ERB-946

NRCC No. 20754

National Research Council Canada
Conseil national de recherches Canada

Division of Electrical Engineering
Division de génie électrique

SEGMENT-MATCHING OF SHAPES BASED ON CLIQUE DETECTION

S. SAKANE AND T. KASVAND

OCTOBER 1982

40

Canada

Copyright 1982 by
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables from this report, provided that the source of such material is fully acknowledged.

Copyright 1982 par
Conseil national de recherches du Canada

Il est permis de citer de courts extraits et de reproduire des données ou tableaux du présent rapport, à condition d'en identifier clairement la source.

Additional copies are available free of charge from:

Editorial Office, Room 301
Division of Electrical Engineering
National Research Council of Canada
Ottawa, Ontario, Canada
K1A 0R8

Des exemplaires supplémentaires gratuits peuvent être obtenus à l'adresse suivante:

Bureau des publications, Pièce 301
Division de génie électrique
Conseil national de recherches du Canada
Ottawa (Ontario) Canada
K1A 0R8

SEGMENT-MATCHING OF SHAPES BASED ON CLIQUE DETECTION

S. Sakane and T. Kasvand

ERB-946

NRCC No. 20754

OCTOBER 1982

TABLE OF CONTENTS

Page

ABSTRACT/RÉSUMÉ v

1 INTRODUCTION 1

2 APPROACHES TO SEGMENT-MATCHING 2

3 SEGMENT-MATCHING OF SHAPES BASED ON CLIQUE DETECTION 3
- 3.1 Applications of Clique Detection 3
- 3.2 Description of the Segment-Matching Method 4
  - 3.2.1 Construction of the First Graph . . 6
  - 3.2.2 The First Clique Detection 9
  - 3.2.3 Grouping Cliques Using the Second Graph 11
  - 3.2.4 Construction of the Third Graph and the Second Clique Detection 13

4 EXPERIMENTS ON SEGMENT-MATCHING 14
- 4.1 A Scale-Variant Matching Problem: Matching Shapes of Machine Parts 14
- 4.2 A Scale-Invariant Matching Problem: Matching Shapes of a Three-Dimensional Model 17

5 CONCLUSION 25

ACKNOWLEDGEMENTS 25

REFERENCES 25

APPENDIX A: CALCULATION OF FOURIER DESCRIPTORS . 28

APPENDIX B: IMPLEMENTATION OF A CLIQUE DETECTION PROGRAM 30

LIST OF FIGURES

		Page
1	An undirected graph	4
2	Structure of the graphs	5
3	Representation of angles at corners	6
4	Differences of the central directions of corners	7
5	Differences of the distances of corners	8
6	Differences of Fourier descriptors	8
7	Contour of a model	15
8	Contour of a model	15
9	Contour of a model	16
10	Contour of a model	16
11	Contour of overlapping objects	17
12	Contour of overlapping objects	19
13	Contour of overlapping objects	20
14	Contour of overlapping objects	21
15	Matching results for TH5 of (a) 0.00005, (b) 0.00002, (c) 0.00001	22
16	Matching results for NFD of (a) 2, (b) 4, (c) 5	23
A1	Position vectors of corners	29
B1	Binary representation of sets	30
B2	Format of an undirected graph	31
B3	Program listing for clique detection	32
B4	An example of detecting cliques of Fig. 1	46

LIST OF TABLES

1	Johnston's clique detection algorithm	10
2	Pairs of corners obtained in the matching	18

ABSTRACT

Shape matching plays an important role in robot vision. When objects are touching, overlapping or partly outside the field of view of the camera, the contours or outlines can become very complicated. Consequently, contour-segment matching is necessary instead of computing conventional shape measures. This report presents a method of segment-matching for shapes, based on clique detection techniques. The clique detection technique is applied to an association graph which represents candidates of corresponding pairs of shape elements between a real image and the model. Since more than one clique may correspond to the outline of the objects, clique detection is again applied to another graph which represents the relations between the cliques in the association graph. The experimental results show the feasibility of applying the method to real scenes.

RÉSUMÉ

L'appariement de formes joue un rôle important dans la vision automatisée. Lorsque les objets se touchent, qu'ils se chevauchent ou qu'ils se trouvent en partie en dehors du champ de la caméra, les contours ou les silhouettes peuvent devenir très compliqués. Il devient alors nécessaire de procéder par appariement de segments plutôt que par calculs de formes classiques. Le présent rapport fait état d'une méthode de reconnaissance de formes par appariement de segments basée sur des techniques de détection de cliques. La technique de détection de cliques est appliquée à un graphe d'association qui répresente des paires correspondantes possibles d'élements de la forme à reconnaître, entre une image réelle et le modèle. Etant donné qu'il peut y avoir plus d'une clique correspondant à la silhouette des objets, la technique est à nouveau appliquée à un autre graphe représentant les relations entre les cliques dans le graphe d'association. Les résultats expérimentaux montrent qu'il est possible d'appliquer la méthode à des scènes réelles.

SEGMENT-MATCHING OF SHAPES BASED ON CLIQUE DETECTION

S. Sakane[1] and T. Kasvand

1. INTRODUCTION

The problem of shape recognition has various levels when dealing with the matching between a real image and its prototype, or the model. A simple example is to classify shapes with respect to their global shape measures such as moment invariants, Fourier descriptors or other geometric features (Pavlidis 1980). If these feature measures are calculated for the object, the problem becomes simply that of classification in the feature space. That is, the distance from the models in the feature space will give us a good cue to identify the object.

However, we must deal with a more complex matching problem, especially in the case of robot vision. For instance, we cannot expect normal outline shapes when the objects are touching or overlapping each other. Their shapes may be partly missing when they are outside the field of view of the camera. These irregular conditions make it impossible to use global shape measures which usually require perfect outline shapes. Consequently, we need segment-matching methods (Davis 1979) in which a segment of the shape extracted from a real image is examined for match to a segment of the shape of the model. In recent years, much attention has been paid to structural approaches to overcome the defects of global measure approaches.

In structural matching some elements, or primitives, of shapes are defined and their relationships must be taken into account in the matching process. Many techniques have been developed recently. Syntactic methods have been applied to shape matching (You and Fu 1979, 1980, Pavlidis and Ali 1979, Pavlidis 1979). Relaxation schemes are also a remedy to solve ambiguous labeling in shape matching (Davis 1979, Rutkowski et al. 1981, Cheng and Huang 1981). There are other proposed techniques based on the comparisons of curvature functions of shapes (Perkins 1978, Wallace et al. 1981).

We present in this paper a new method for segment-matching for shapes based on clique detection techniques. A clique is defined as a maximal complete subgraph. Therefore, the method has the advantage of detecting quite efficiently the maximal set of matched elements. We perform the clique detection twice.

First, we construct an association graph in which a node corresponds to a possible pair of corners between a shape in the real image and a shape in the model. An edge is connected between two nodes when

[1] *Visiting research scientist from the Electrotechnical Laboratory, Automatic Control Division, 1-1-4 Umezono, Sakuramura, Niihari-gun, Ibaraki 305, Japan.*

they are consistent with respect to some criterion. By changing this criterion, the method can be used for both scale-variant and scale-invariant matching problems, depending on the applications. A clique in the first graph implies a maximal consistent pair of corners. Since more than one clique may correspond to the shape of the object, we should examine whether two or more cliques can be merged into one. To achieve this, we construct another graph which represents the relationships among the cliques detected in the first graph. The consistency between two nodes in this graph is defined as being the error in the geometric transformation between the two segments of shapes. Then, clique detection is performed on this graph and the best matched pairs of corners are determined. The method was applied to two problems, the matching of shapes of touching or overlapping machine parts and the matching of silhouettes of a three-dimensional model. In the latter problem, scale-invariant matching is required since the shape of a real image and the model have different scales. The experimental results show the feasibilty of applying the method to these problems.

2. APPROACHES TO SEGMENT-MATCHING

In recent years, many structural approaches have been developed for segment-matching of shapes. We briefly review those approaches.

Syntactic methods use grammars, or production rules, to describe the relationships among the primitive shapes. You and Fu (1980) used an error-correcting technique combined with primitive-extraction-embedding parser. Their method was tested on distorted shapes of airplanes when they are partially occluded by a cloud. Pavlidis and Ali (1979) developed a syntactic analyzer. It accepts polygonal approximations of contours and produces a sequence of arcs, each with such attributes as relative size, arc type and orientation. For instance, the arc types are corner, quadratic curve, sharp protrusion, sharp intrusion, etc. Pavlidis (1979) demonstrated that the analyzer can be successfully applied to the matching of contour data of coast lines. Generally syntactic approaches can deal with context based on shape information to a certain degree. However, the major disadvantage of syntactic methods is that they cannot avoid difficulties in generating grammars when the complexity of shapes is increased.

Relaxation schemes have been used recently to solve ambiguous labeling problems. Davis (1979) applied the discrete relaxation to the association graph which represents candidate pairs of angles between an object and a template. Finally, matched pairs are obtained in the graph using an ordered search technique. The experiments were performed on the shape data of coastlines. Since this algorithm has scale-dependent characteristics, a scale-invariant matching method combined with Fourier descriptors was developed (Sakane 1981). Rutkowski et al. (1981) applied probablistic relaxation to prune a directed graph, where a node corresponds to a piece of the object boundary. A directed edge in a graph means that the two segments may be consecutive along the boundary of an object. A probability that an extracted segment corresponds to a part of a model

segment is defined. These probabilities are updated using a relaxation scheme. In their experiments, however, the model shape of an airplane was segmented only into four pieces, nose, left wing, right wing and tail. Therefore, it was easy to establish the relationships among segments.

Perkins (1978) developed a model-based vision system which determines the position and orientation of curved objects. The edge elements were extracted and linked to form chain structures. They were then fitted to straight lines or circular arcs or combinations of both called concurves. The matching process has three phases. First, general properties of image and model concurves are compared. Then, a tentative transformation from model to image coordinates is determined by using one combination. Finally, a set of points equally spaced along the model concurves is transformed to image space in order to determine whether the tentative transformation is acceptable or not.

Wallace et al. (1981) proposed a matching technique based on the comparison of curvature functions. They defined feature vectors in terms of curvature maxima and minima which correspond to peaks in the derivative function of the curvature. The feature vectors were compared between a real contour of the object and library models. The experimental results showed good performance even when compared to the global method of Fourier descriptors.

Each of these methods have certain characteristics for selecting elements and for processing the relationships among the matched elements. Thus, they may be suitable for some applications but not for others.

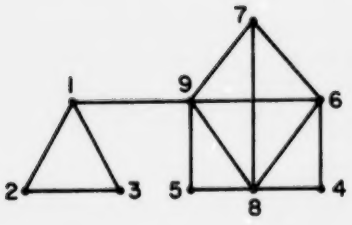
3. SEGMENT-MATCHING OF SHAPES BASED ON CLIQUE DETECTION

We propose a method of segment-matching using graph processing techniques. Since we cannot expect exact matching in general circumstances, the system should produce as many matching pairs of elements as possible. Clique detection technique is one of the remedies to achieve this purpose. A clique is defined as a maximal complete subgraph of an undirected graph. Therefore, every two nodes in a clique have an edge between them and there exists no larger complete subgraph which includes the clique. For example, the graph in Fig. 1a has cliques shown in Fig. 1b.

If we have a graph where each node represents a possible matched pair of elements and each edge is defined to express consistency between two nodes, cliques in the graph give the candidate sets of matched pairs. We first review some prior work on the application of clique detection techniques to computer vision. Then we describe a new method for segment-matching of shapes.

3.1 Applications of Clique Detection

The first attempt can be seen in the robot vision system developed by Ambler et al. (1975). They employed clique detection tech-



a

(1, 9)
(1, 2, 3)
(4, 6, 8)
(5, 8, 9)
(6, 7, 8, 9)

b

Fig. 1 (a) An undirected graph.
(b) Cliques of the graph in (a).

niques to recognize objects in their versatile assembly system. Since the objects in the experiments had simple shapes, it was easy to segment a shape into corners and sides which were used as primitives in matching. Clique detection was used three times; to match segments in an outline shape, to match holes in a region, and to match objects on a table top.

Bolles applied a clique detection technique to locate features such as holes in machine parts (Bolles 1979). The relations between holes were the types of holes (actually the size), and the distance between two holes. However, the outlines of the shapes of the objects were not taken into account.

Radig et al. (1980) also used clique detection to match line features between a real image and the model. The matching elements are pairs of straight lines and regions. In case of straight lines, for example, the relations among the elements are simply based on the consistency in the positions of end-points and the angles.

Cheng and Huang (1981) proposed a maximal weighted clique detection method for image matching using a relaxational scheme. They employed a 'star structure' which contains a possible master mapping and several associated mappings, instead of using the conventional node structure. A compatibility measure between each two star pairs was computed from the number of consistent mappings between these star pairs. The probabilities of star pairs are updated in a relaxational manner. The star pair with the highest probability gives the final match. They gave an application example to recognize airplanes in aerial imagery.

When we use clique detection techniques, we have to note that (i) the size of the graph should be limited and as small as possible since the algorithm has exponential complexity (Aho et al. 1974), and (ii) the largest clique does not always correspond to the match we expect (Bolles 1979).

3.2 Description of the Segment-Matching Method

We use three graphs called the first, the second and the third graph (Fig. 2). These graphs are constructed in this order. We apply

clique detection on the first and the third graph. The first graph is also called an association graph (Davis 1979) in which a node corresponds to a possible pair of angles between those in the real image and the model. An edge is connected between two nodes when they satisfy some conditions of consistency. Therefore, cliques in the first graph mean maximaly consistent associated pairs. In order to classify the cliques of the first graph, we then construct the second graph where each node corresponds to a clique of the first graph and we extract connected

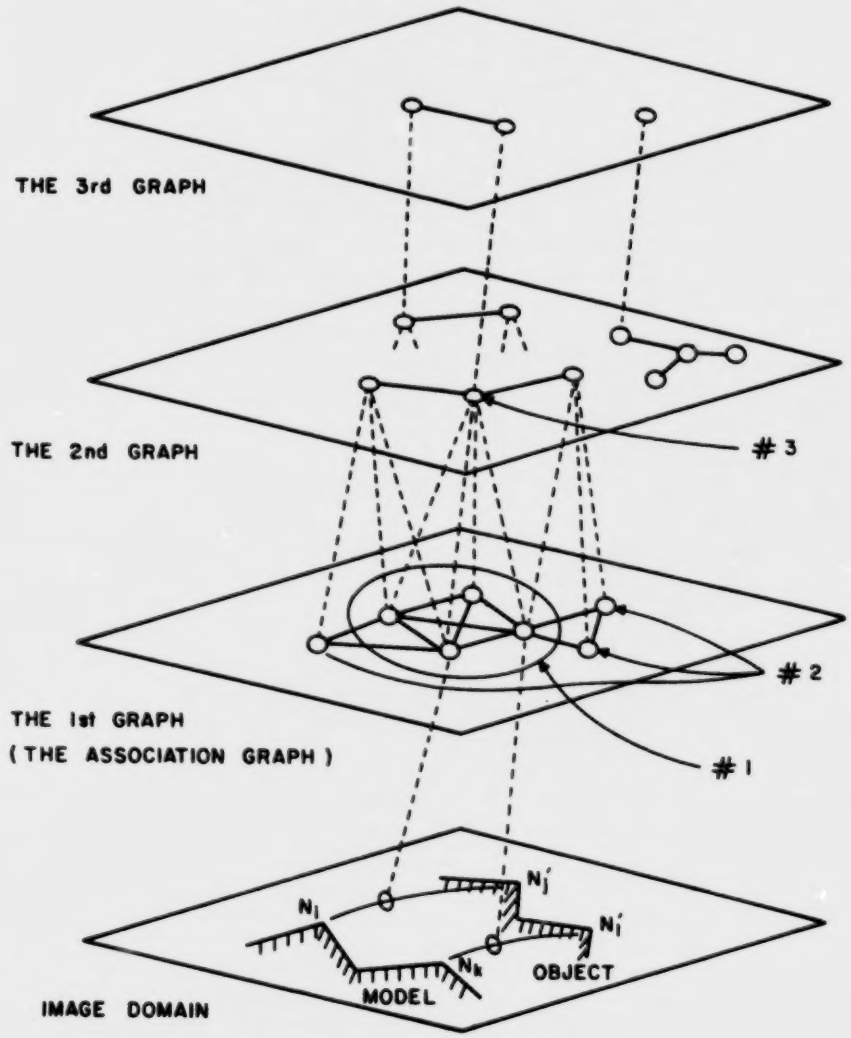


Fig. 2 Structure of the graphs. (#1) Corner pairs used to define the coefficients of the geometric transformation between object and model. (#2) If corner pairs corresponding to these nodes are not yet in kernel node, test geometric transformation error. (#3) The kernel node (for example).

components in the second graph. A kernel node is defined in each connected component. Since more than one clique of the first graph may correspond to the model shape, we must examine the relationships of the kernel nodes in the second graph. To achieve this systematically, we construct the third graph. The third graph represents the kernel nodes of the second graph and the relationships among them. An edge is connected between two nodes if the two sets of pairs are consistent with respect to their geometric transformation. Therefore, cliques of the third graph are final candidates of the matching. The set of pairs with the highest score gives the best candidate for matched pairs. The details are described in the following sections.

3.2.1 Construction of the First Graph

The matching elements, or primitives, of our method are the corners in contours of shapes, i.e., the same as in Davis (1979). Each element has (x, y) coordinates and a bending angle at the corner (Fig. 3). We assume that the shapes are already approximated by polygonal curves or the corner points are detected by some other methods, for instance, based on the curvature function or the chain code expression (Hung and Kasvand 1982). First, all pairs of corners between the object and the model with similar angle are associated and registered as nodes in a graph. We call this graph the first graph or the association graph. The condition for association of two corners N_i and N_j' is

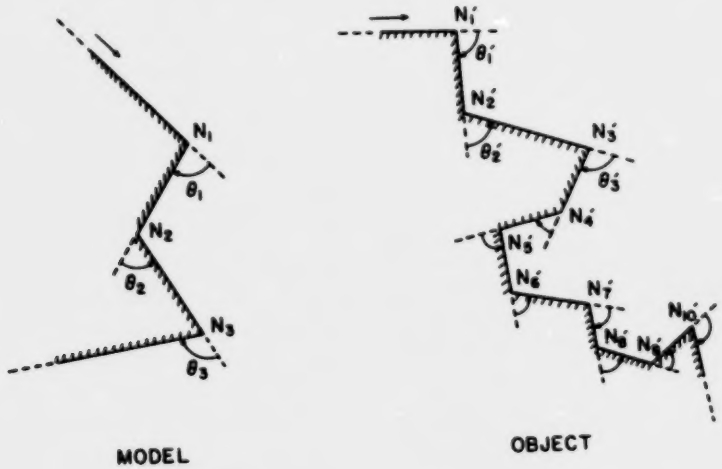


Fig. 3 Representation of angles at corners. N_i and N_j' are clockwise sequential label numbers and $i = 1,2,\ldots,m$, $j = 1,2,\ldots,n$. Examples of nodes of the first graph (association graph) are (N1,N1') = node#1, (N1, N7') = node#2, (N2,N2') = node#3, (N2, N6') = node#4, (N2, N2') = node#5, (N2,N9') = node#6, (N3, N3') = node#7, (N3,N10') = node#8, etc.

$$
\begin{aligned}
&| \theta_i | > TH1 \\
[1] \qquad &| \theta_j' | > TH1 \\
&| \theta_i - \theta_j' | < TH2
\end{aligned}
$$

where $i = 1,2,\ldots,n$, $j = 1,2,\ldots m$, θ_i and θ_j' are their bending angles and n and m are the number of corners in the model and object, respectively. TH1 and TH2 are thresholds. After all possible nodes are generated, we then determine whether or not two nodes should be connected by an edge, in order to represent the relationships among nodes. The criterion to connect two nodes is to satisfy the conditions (a) to (d) below:

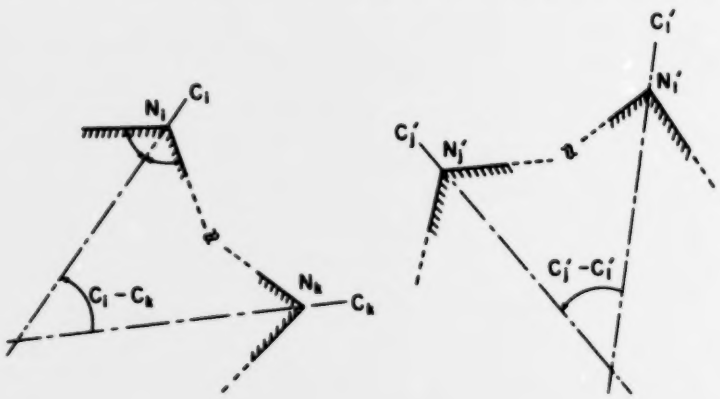
(a) Two corners must satisfy the sequencial labeling order. Each corner has a label numbered clockwise with respect to the objects. When the two pairs of corners are expressed as (N_i, N_j') and (N_k, N_l'), the conditions are

[2] $\quad N_j' > N_l'$ (if $N_i > N_k$), $\qquad N_j' < N_l'$ (if $N_i < N_k$)

(b) Two corners must have a mutual relationship with respect to the central directions of the corners (Fig. 4). That is, when the corner N_i has the direction C_i and the other corner N_j' has the direction C_j', the condition is

[3] $\quad | (C_i - C_k) - (C_j' - C_l') | < TH3$

(c) Two corners must have a relationship with respect to the



MODEL OBJECT

Fig. 4 Differences of the central directions of corners. N_i, N_k, N_j', and N_l' are clockwise sequential corner labels on the model and object, respectively. C_i, C_k, C_j', and C_l' are the corresponding "central directions".

distance between them (Fig. 5). Therefore, when the corners i, j, k, l have the position vectors P_i, P_j, P_k, P_l

$$[4] \qquad \left| \left| P_i - P_k \right| - \left| P_j' - P_l' \right| \right| < TH4$$

(d) The contour segments between the two respective corners must have a similarity in terms of Fourier descriptors (Sakane 1981). That is, when A(n) and B(n) are the nth terms in the Fourier expansions of the partial shapes (Fig. 6 and Appendix A),

$$[5] \qquad \sum_{n=1}^{NFD} \left(\left| A(n) \right| - \left| B(n) \right| \right) < TH5$$

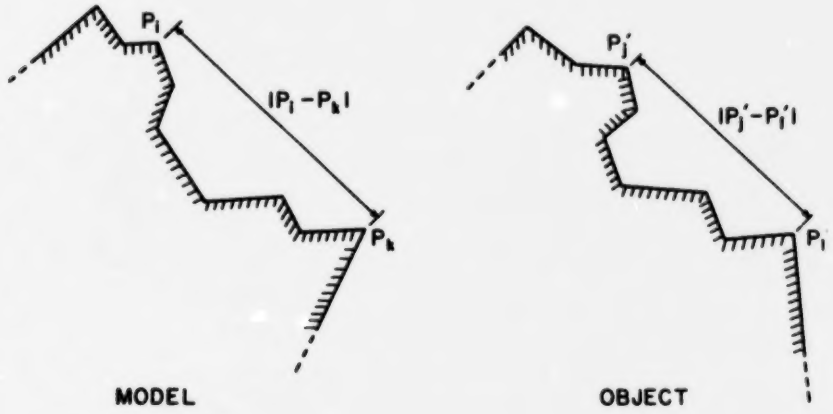


Fig. 5 Differences of the distances of corners. P_i, P_k, P_j', and P_l' are position vectors on model and object, respectively.

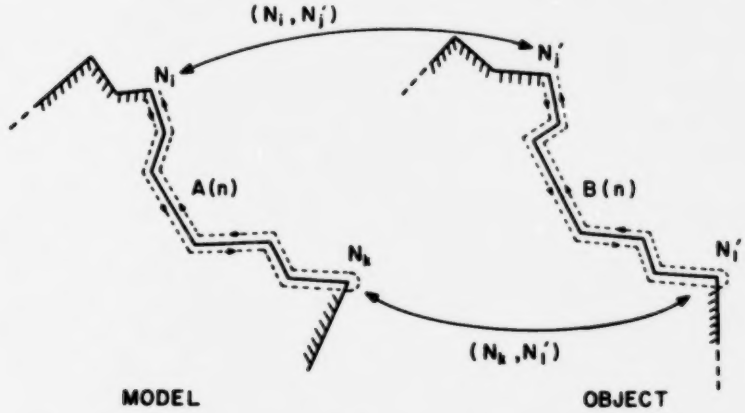


Fig. 6 Differences of Fourier descriptors. N_i, N_k, N_j', and N_l' are corner labels. A(n) and B(n) are the nth components of the Fourier expansions.

The number NFD is a maximum number of terms of the Fourier expansion to be considered in the measure. This number affects the discriminant characteristics of shape similarity in the frequency domain. Therefore, it should be determined according to the variations of contour shape in given applications. The calculation of the Fourier descriptors is shown in Appendix A.

The condition (a) may generate a problem when the shape is a closed curve and the number sequence begins from the middle part of one segment shape but not in the other segment shape. For example, even if a set of pairs ((2,18), (3,19), (4,20), (5,1), (6,2), (7,3)) is the best match, the first three pairs are not connected to the last three pairs, due to condition (a). Therefore the first three pairs and the last three pairs are obtained as different cliques in the association graph. We may avoid this problem if we can renumber the corners of one shape properly. The Algorithm F proposed by Davis (1979) has the same problem since the condition (a) is used in the discarding process of the nodes in the association graph.

In our method, however, we can merge the set ((2,18), (3,19), (4,20)) and the set ((5,1), (6,2), (7,3)) in the second clique detection process described in a later section.

Since the condition (c) is scale dependent, we may ignore this condition when scale invariance is necessary in an application. This example will be shown in the experiments.

3.2.2 The First Clique Detection

After the construction of the association graph, we extract cliques in the graph. Many algorithms for clique detection have been proposed from the graph theoretical viewpoint (Bron and Kerbosch 1973, Johnston 1976, Gerhards and Lindenberg 1979). We used Johnston's clique detection algorithm (Johnston 1976). The algorithm is briefly described below.

The algorithm is shown in Table 1. The recursive function EXPAND has three sets of arguments; COMBINATION, INTERSECTION and POSSIBLE. COMBINATION is a set in which a clique is constructed step by step taking an element from a set POSSIBLE. Every two nodes in a clique must be connected. Therefore, when we put a node J into COMBINATION, we can remove from POSSIBLE all nodes which are not connected to the node J. INTERSECTION contains a set to determine whether the current set in COMBINATION is a clique or not. When INTERSECTION becomes an empty set, there is no node connected to all nodes already in COMBINATION. In other words, the COMBINATION contains a clique. The program starts with the initial sets,

COMBINATION = {empty set}
INTERSECTION = {all nodes in the graph}
POSSIBLE = {all nodes in the graph}

Table 1. Johnston's clique detection algorithm (Johnston (1976)

```
procedure FIND AND USE ALL CLIQUES( value set V;
                    value graph G; procedure USE );
begin
    procedure EXPAND(value set COMBINATION,
                                INTERSECTION,POSSIBLE)

    if INTERSECTION=empty, then USE(COMBINATION) else
    begin set S;
      CALCULATE S;
      while (S is not empty) do
      begin integer I;
        I := any member in S
        S := S - ( I )
        POSSIBLE := POSSIBLE - ( I )
        EXPAND((COMBINATION U ( I )),
                INTERSECTION∩ G(I), POSSIBLE∩ G(I) )
      end;
    end EXPAND;
    EXPAND( (), V, V )
end FIND AND USE ALL CLIQUES;


procedure CALCULATE S;
  S := POSSIBLE - G( any member in INTERSECTION )
```

The algorithm can cut off many useless branches in the search tree. It is explained as follows. Assume we select an element J of INTERSECTION. We can separate the set S (= POSSIBLE) into two sets. That is,

$$S1 = S - G(J)$$
$$S2 = S - S1 = S \cap G(J)$$

where G(J) denotes the set of adjacent nodes of a node J. G(J) does not include the node J itself. We use (-) to denote subtraction of sets. When we have finished processing S1 first, the set POSSIBLE for the further calls for EXPAND is,

$$POSSIBLE - S1 = S - S1 = S2$$

Therefore, every node I in COMBINATION must belong to S2. Since S2 is a subset of G(J), node I belongs to G(J). This implies that the node J belongs to G(I). Thus,

$$J \in INTERSECTION \cap G(I)$$

This condition means that the node J always remains in the further calls

to EXPAND after processing S1. Therefore, no new cliques will be found in the further calls. We can reduce the useless branches of the search tree by calculating the set S as

$$S = POSSIBLE - G(J)$$

It might be advantageous to select the node J which has many adjacent nodes, to cut off as many useless branches as possible. The algorithm can be modified by changing the part for calculating S. However, the process of comparisons to select such proper nodes becomes an overhead. Johnston (1976) also proposed another method to calculate the set S. The algorithm is equivalent to the Bron-Kerbosch algorithm (Bron and Kerbosch 1973) in its special case. The conclusions of his experiments were that the first method is the best for applications to low density graphs, while it is worthwhile to use the second method for higher density graphs, though the terms 'lower' and 'higher' depend on the details of the implementation and the hardware of the computer used. We employed the algorithm called the simplified Bron-Kerbosh algorithm (Johnston 1976), in which the node J is selected arbitrarily. The details of the program are explained in Appendix B.

The algorithm can extract all cliques in a graph. However, it is not efficient to store all cliques for further processing even if the clique has only a small number of elements. Therefore, we restrict the minimum size of a clique to be used for matching. In our experiments, the minimum size was three and the cliques with less than three elements were ignored.

3.2.3 Grouping Cliques Using the Second Graph

Since a clique has a characteristic that it is easily split into two cliques if one edge is missing, we will detect many similar cliques that have almost identical node sets. In addition, it is difficult to adjust the thresholds TH1 to TH5 so that all correct association pairs are included just in one clique. In order to reduce the candidate cliques, we classify the cliques into groups and select one candidate clique for each group. We then expand the candidate cliques to obtain more matched pairs within the group, if desired. This process is performed as follows (Fig. 2):

(1) Construct a graph, called the second graph, in which each node corresponds to a clique in the first graph. When two cliques in the first graph have at least one common node, we connect an edge between the corresponding two nodes in the second graph.

(2) Detect connected components in the second graph. A connected component is a maximal set of nodes in which every two nodes are connected, i.e., a set of connected nodes.

(3) Determine a kernel node in each connected component. We define the kernel node as the one which has the maximum score in a

connected component. The score of a node is calculated as the sum of scores for each corner which is a member of the pairs of the node. That is, when a node has associated pairs; $(N_i(m), N_j'(m))$, $m = 1,\ldots,M$ and the (x, y) coordinates of a corner N_i are $(X(N_i), Y(N_i))$, the score of the node is defined as,

$$[6] \qquad SCORE = \sum_{m=1}^{M} (LENGTH\ (N_i(m), N_i(m) - 1) + LENGTH\ (N_i(m), N_i(m) + 1))^* \left| ANGLE\ (N_i(m)) \right|$$

where $ANGLE(N_i(m))$ is the absolute value of the bending angle at the corner $N_i(m)$, and

$$LENGTH\ (N_i, N_k) = SQRT\ (DX^*DX + DY^*DY)$$

$$DX = X(N_i) - X(N_k)$$

$$DY = Y(N_i) - Y(N_k)$$

(4) For each kernel node, examine the associated pair of corners which is not in the kernel node but is in a node of the connected component. If the pair satisfies the condition with respect to the error of the geometric transformation determined by the kernel node, the pair can be merged into the kernel node. The geometric transformation is a combination of translation, rotation and scale change. Assume a kernel node has the pairs of association; $(N_i(m), N_j'(m))$, $m = 1,\ldots,M$, and the corners $N_i(m)$, $N_j'(m)$ have the (x, y) coordinate values; $(X(N_i(m)), Y(N_i(m)))$ and $(X(N_j'(m)), Y(N_j'(m)))$, respectively. We can calculate the parameters of the transformation from the model to the objects as follows:

$$X(N_j'(m)) = a^*X(N_i(m)) - b^*Y(N_i(m)) + c$$

$$Y(N_j'(m)) = b^*X(N_i(m)) + a^*Y(N_i(m)) + d$$

At least two association pairs are necessary to determine the parameters (a, b, c, d). When the number M is greater than 2, we can calculate the parameters using the least-squares solution which minimizes root-mean-square errors. Therefore, when the pair of association (N_k, N_l') is to be examined, the error can be evaluated using these parameters.

$$ERRORx = a^*X(N_k) - b^*Y(N_k) + c - X(N_l')$$

$$ERRORy = b^*X(N_k) + a^*Y(N_k) + d - Y(N_l')$$

The criterion for merging a pair of corners with the kernel node is,

$$[7] \qquad SQRT\ ((ERRORx)^{**}2 + (ERRORy)^{**}2) < TH6$$

where TH6 is a threshold value.

One problem in this merging process is its order dependence. If a pair (N_i, N_j') is already merged into the kernel, we will not allow the merge of another related pair, for example (N_i,N_1'), since the corner N_i is already a member of the pairs in the kernel. Therefore, we first construct a set of corner pairs which are not contained in the kernel node, i.e., they are the remaining nodes of the connected component in the first graph. Then the pairs (remaining nodes) are sorted according to the error values of the geometric transformation. The pair which has the least error value is considered first. If both of the corners in the pair are not yet contained in the kernel node and the error value of this pair is less than the threshold TH6, then the pair is allowed to be merged into the kernel node.

3.2.4 Construction of the Third Graph and the Second Clique Detection

After merging cliques in each connected component of the first graph, we will have many candidate sets of corner pairs corresponding to the contour segments between a real image and the model. However, more than one set of corner pairs may give the best match. In order to check the consistency among the sets of pairs, we construct another graph, called the third graph, and apply clique detection to this graph. In the third graph, a node represents a set of corner pairs obtained in the merging process of Sect. 3.2.3. Every two nodes in the third graph are examined whether the two sets of the association pairs are geometrically consistent or not. The criterion of the consistency is similar to the merging process in Sect. 3.2.3, point (4). Assume, for example, that one node of the third graph has the association pairs (N_i(m), N_j'(m)), m = 1,...,M1, and the other node has the pairs (N_k(n), N_1'(n)), n = 1,...,M2, where M1 > M2. We first calculate the parameters of the geometric transformation with respect to the data of (Ni(m), Nj'(m)). That is,

$$X(N_j'(m)) = a*X(N_i(m)) - b*Y(N_i(m)) + c$$

$$Y(N_j'(m)) = b*X(N_i(m)) + a*Y(N_i(m)) + d$$

give the equations to get the 4-tuple of parameters (a, b, c, d). Then we evaluate the error of each association pair (N_k(n), N_1'(n)) in terms of the transformation described by these parameters. The errors in x and y coordinates are given as,

$$ERRORx(n) = a*X(N_k(n)) - b*Y(N_k(n)) + c - X\ (N_1'(n))$$

$$ERRORy(n) = b*X(N_k(n)) + a*Y(N_k(n)) + d - Y\ (N_1'(n))$$

The condition for connecting two nodes in the third graph is expressed as

$$[8] \qquad \frac{1}{M2} \sum_{n=1}^{M2} \text{SQRT } ((\text{ERRORx}(n))^{**}2 + (\text{ERRORy}(n))^{**}2) < \text{TH7}$$

After the construction of the third graph, the clique detection is performed again. We then take union of the association pairs within a clique. These sets of association pairs are the final candidates. We calculate a score for each candidate set. The score is the same as that used in the selection of the kernel nodes in the connected components of the second graph. The set with the highest score gives the best match.

4. EXPERIMENTS ON SEGMENT-MATCHING

The segment-matching methods described in the previous sections were applied to two matching problems. One is the matching of outline shapes of machine parts and their models where scale-variant matching is used. The other problem is the matching between a silhouette shape of an object and many silhouettes obtained from the three-dimensional model. In this problem, a scale-invariant matching is required since the shapes have different scales.

4.1 A Scale-Variant Matching Problem: Matching Shapes of Machine Parts

We first applied our method to matching shapes of machine parts when the parts are touching or overlapping each other. We assume that the model shapes are extracted from related images taken under similar conditions as the images of the objects. Therefore, scale-variant matching can be applied. The image data were taken through an ITV camera and digitized into 8-bit gray levels. The image size is 256 × 256 pixels. The gray level data were thresholded and converted into binary data using a discriminant thresholding method (Otsu 1979). Then the boundaries of regions were extracted and approximated by straight-line segments based on a splitting algorithm (Ramer 1972). The corners of the boundaries are labeled by a sequence of numbers. Figures 7 to 10 show the models to be matched. Figures 7a to 10a show the boundaries of a regions after thresholding. Figures 7b to 10b are the approximated boundaries with the tolerance of 2.0 pixel length. The numbers put along the boundary show the labels of the corners. Figures 11a and b show boundaries of overlapping objects. The match between the shape in Fig. 7b and the shape in Fig. 11b is shown in Fig. 11c. The circles indicate the corners obtained as matched pairs. Table 2 shows the matched pairs of corners. The numbers on the left are the corners of the model (Fig. 7b) and the numbers on the right are the corners of the objects (Fig. 11b).

In this example, two cliques were merged in the third graph even though they were obtained as disjoint cliques in the first graph. The number of nodes generated in the first graph was 324 and the number of edges was 116. The processing time was about 6 s on VAX11/780, excluding the time for extraction of contour data. Most of the time was spent generating the first graph. The clique detection was performed within 1 s. Figure 11d shows the match between Fig. 8b and Fig. 11b. Figures

12 to 14 show the other results. All the objects were correctly identified. In these experiments, the parameters used in the matching were

TH1 = 0.12 rad	TH5 = 0.00005
TH2 = 0.75 rad	NFD = 3
TH3 = 0.75 rad	TH6 = 20.0 pixels
TH4 = 25.0 rad	TH7 = 20.0 pixels

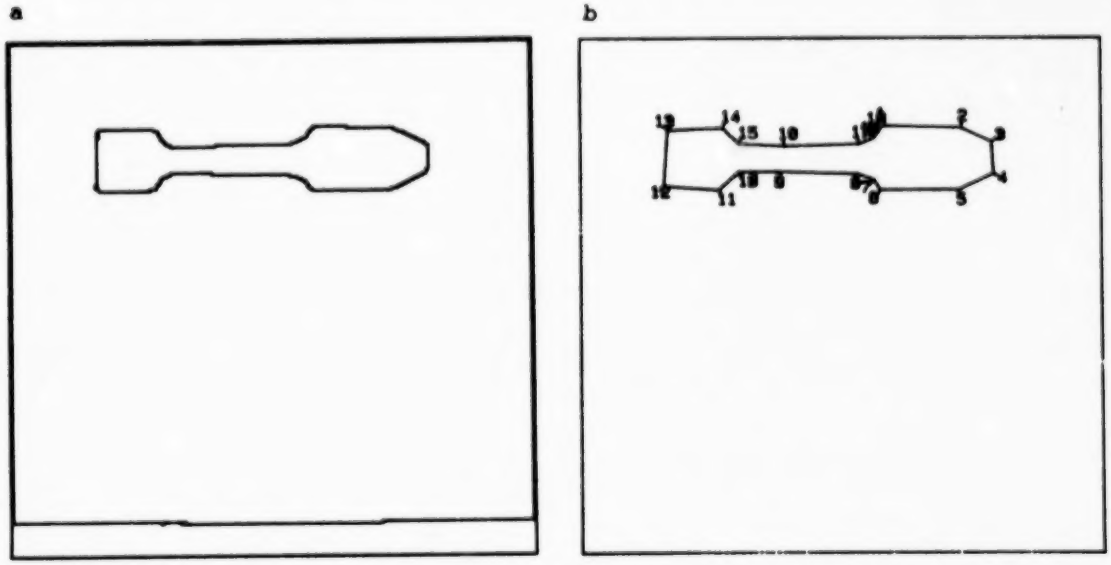


Fig. 7 (a) Contour of a model. (b) Polygonal approximation of (a).

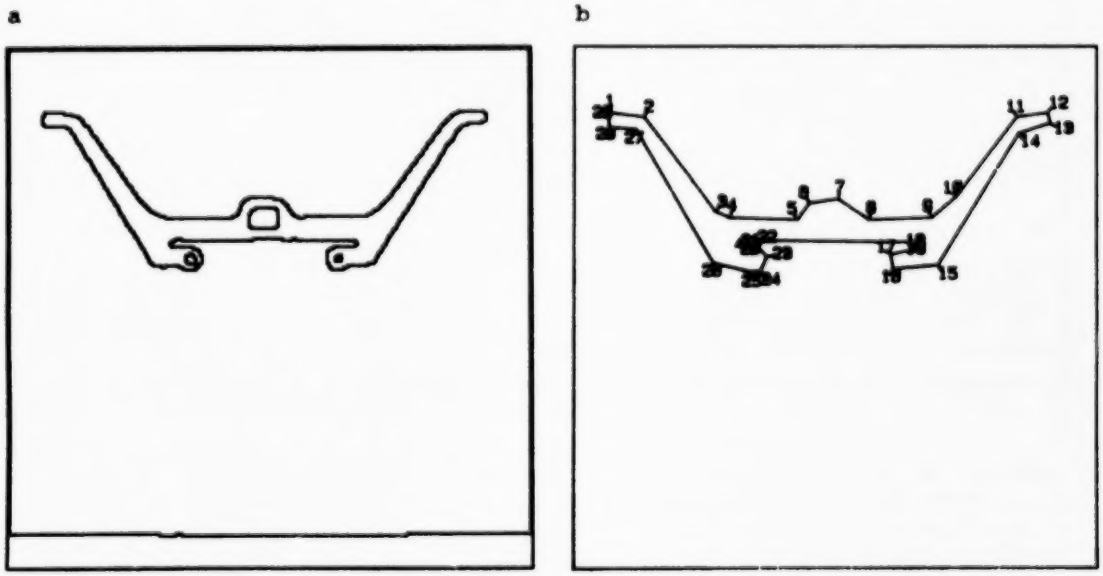


Fig. 8 (a) Contour of a model. (b) Polygonal approximation of (a).

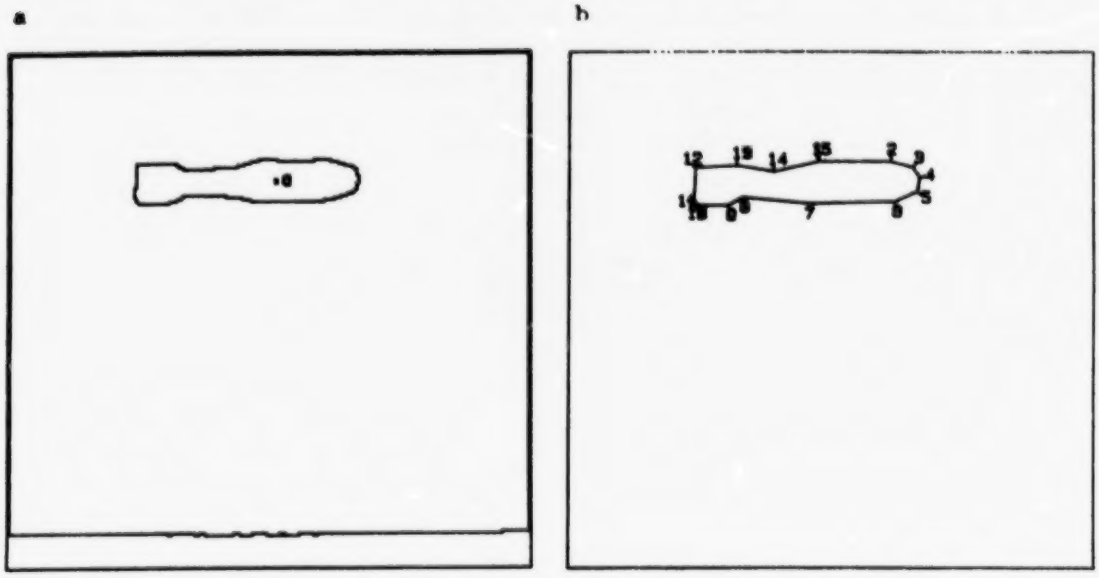


Fig. 9 (a) Contour of a model. (b) Polygonal approximation of (a).

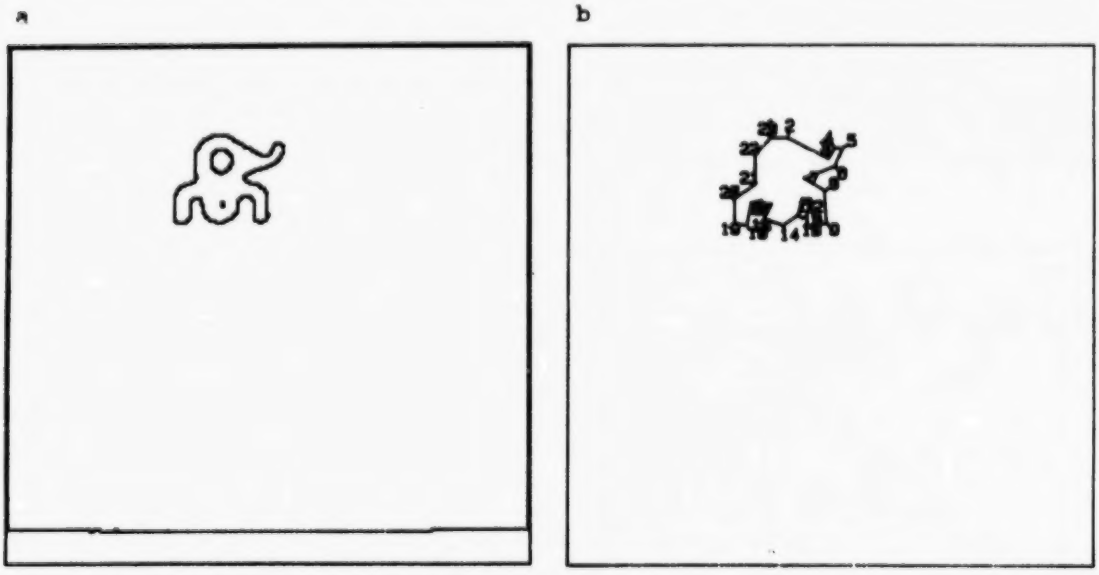


Fig. 10 (a) Contour of a model. (b) Polygonal approximation of (a).

Since some of the parts had thickness, the contour shapes did not always produce unique corners due to lighting conditions. In addition, the program for polygonal approximation does not locate unique positions for the corners. However, despite these problems, the experimental results were sufficient to show the robust character of this matching method.

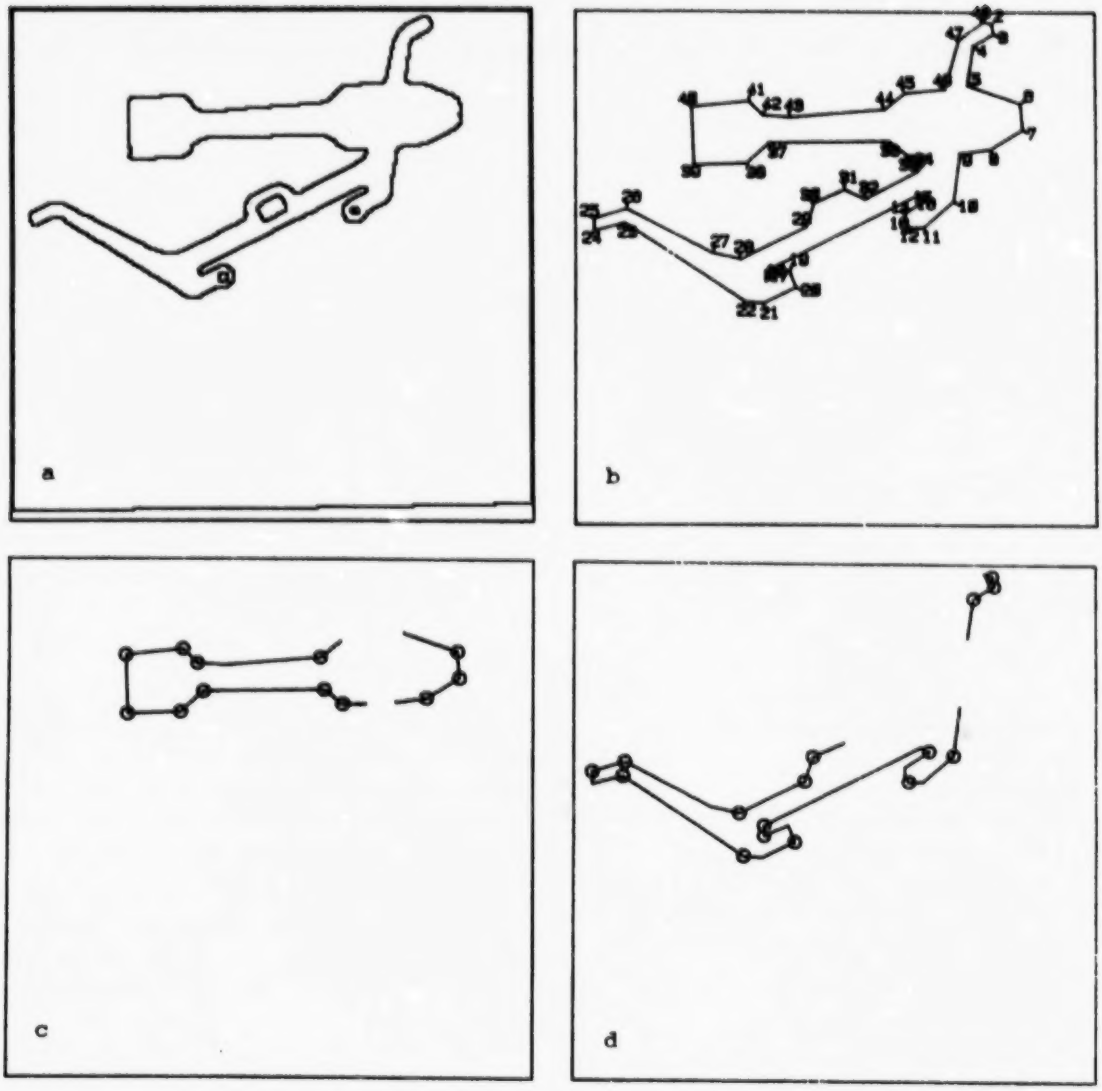


Fig. 11 (a) Contour of overlapping objects. (b) Polygonal approximation of Fig. 11a. (c) Results of matching Fig. 7b and Fig. 11b. (d) Results of matching Fig. 8b and Fig. 11b.

4.2 A Scale-Invariant Matching Problem: Matching Shapes of a Three-Dimensional Model

The data of this experiment are the same as in Sakane (1981). The problem is to estimate, given a real image, the position of the viewpoint to this object. That is, shape matching is required between a silhouette of the real image and one of the model silhouettes generated by changing the position of the viewpoint to the three-dimensional model.

Table 2. Pairs of corners obtained in the matching.

ITEM	ITEM
(3 6)	(1 25)
(4 7)	(2 26)
(5 8)	(3 28)
ITEM	(5 29)
(6 35)	(6 30)
(8 36)	ITEM
(10 37)	(12 2)
(11 38)	(13 3)
(12 39)	(14 4)
(13 40)	ITEM
(14 41)	(15 10)
(15 42)	(17 12)
(16 43)	(18 15)
(17 44)	(20 17)
	(21 18)
	(24 20)
	(26 22)
	(27 23)
The results for Fig 11,(c).	The results for Fig 11, (d).

In the silhouette of the real image, the length is scaled by pixel. On the other hand, in the silhouette of the model, the scale is determined by a combination of the actual length of the model and the camera parameters which are set at a specified viewpoint. Thus, they have different scales and, therefore, a scale-invariant matching is required.

The viewpoints were assumed to be on a circle near the ground with fixed diameter from a specified center in the model. Thirty-six line-drawing models were generated, by changing the viewpoint by 10° intervals on the circle. The real scene is a miniature object (physical scaled model) in 1/400 scale, scanned with an ITV camera. The positions of the camera were also set on a circle similar to that of the geometrical mathematical model and the diameter was about the same as the converted value. Thus, the causes of shape variations are restricted to moving the view-point around the object. The image was 256 × 192 in size and digitized in 7-bit gray levels. The silhouette shape was extracted in similar manner as in Sect. 4.1.

The results of matching a silhouette at the 240° position and the 36 model silhouettes are shown in Fig. 15. The parameters used in the matching were,

TH1 = 0.12 rad NFD = 3
TH2 = 0.75 rad TH6 = 5.0 pixels
TH3 = 0.75 rad TH7 = 20.0 pixels

The threshold values for TH5 were,

TH5 = 0.00005 (Fig. 15a)
TH5 = 0.00002 (Fig. 15b)
TH5 = 0.00001 (Fig. 15c)

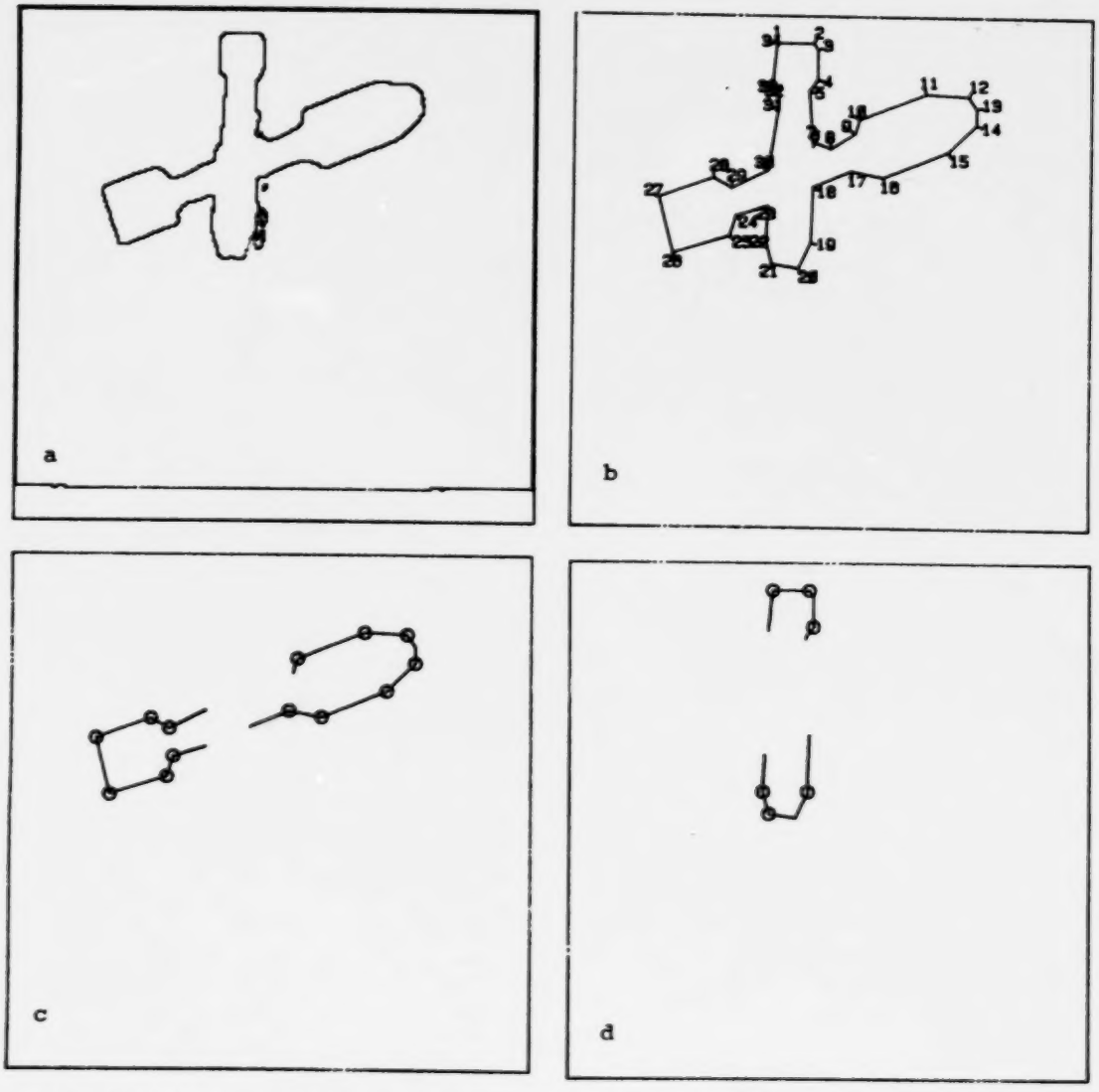


Fig. 12 (a) Contour of overlapping objects. (b) Polygonal approximation of Fig. 12a. (c) Results of matching Fig. 7b and Fig. 12b. (d) Results of matching Fig. 9b and Fig. 12b.

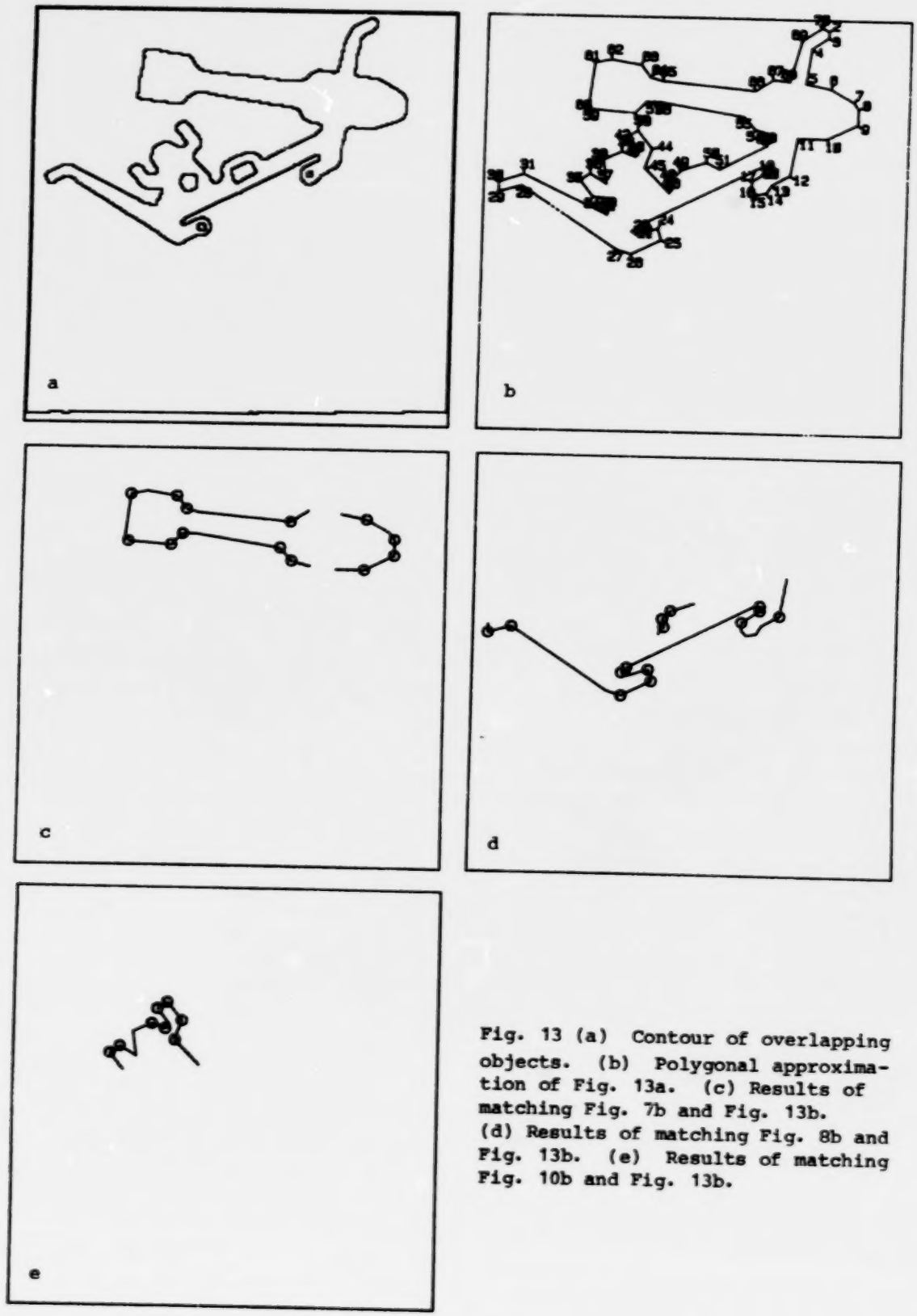


Fig. 13 (a) Contour of overlapping objects. (b) Polygonal approximation of Fig. 13a. (c) Results of matching Fig. 7b and Fig. 13b. (d) Results of matching Fig. 8b and Fig. 13b. (e) Results of matching Fig. 10b and Fig. 13b.

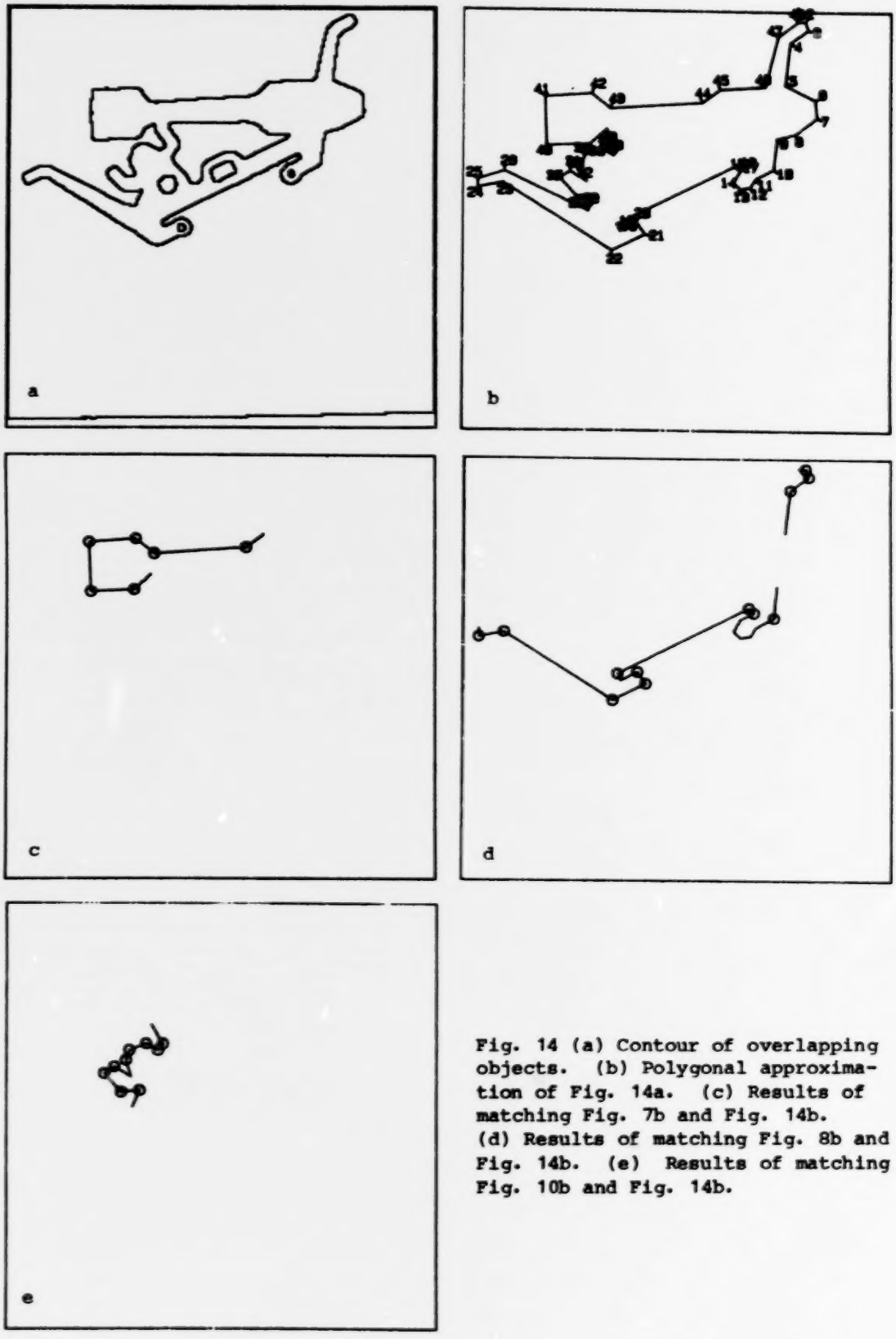


Fig. 14 (a) Contour of overlapping objects. (b) Polygonal approximation of Fig. 14a. (c) Results of matching Fig. 7b and Fig. 14b. (d) Results of matching Fig. 8b and Fig. 14b. (e) Results of matching Fig. 10b and Fig. 14b.

a

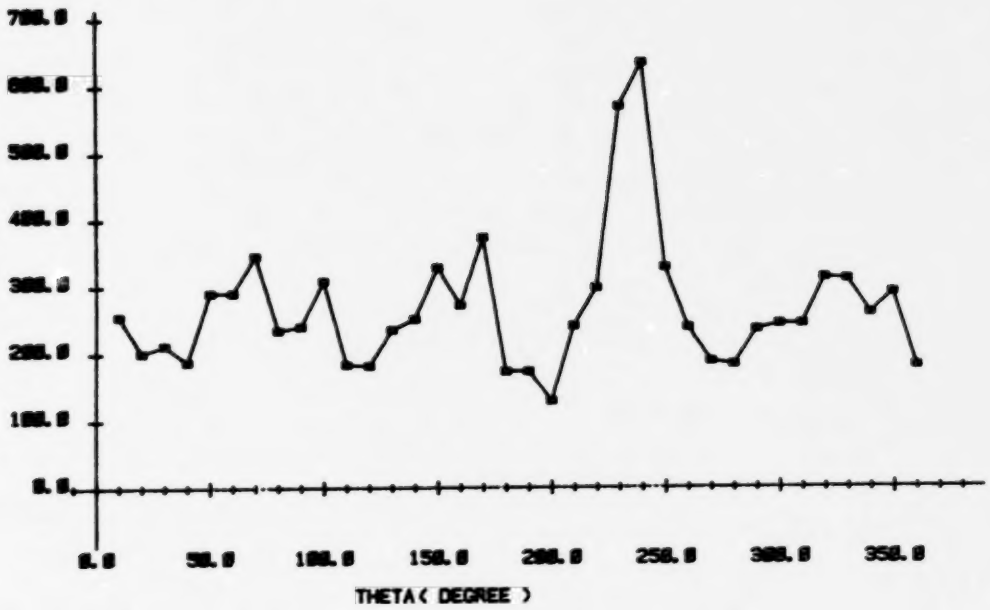


Fig. 15 (a) Matching results for TH5 = 0.00005.

b

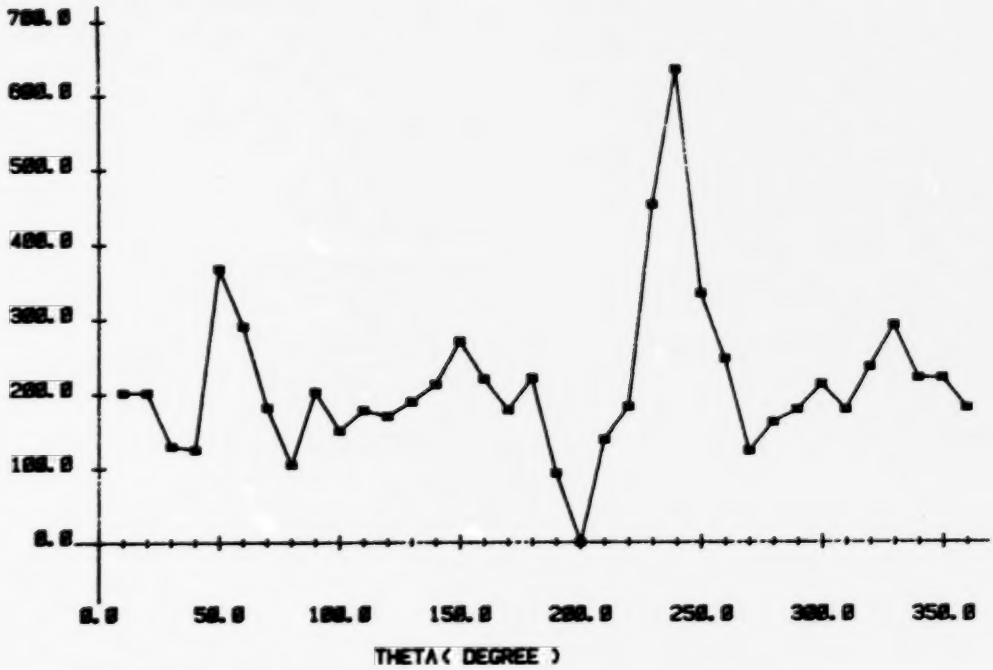


Fig. 15 (b) Matching results for TH5 = 0.00002.

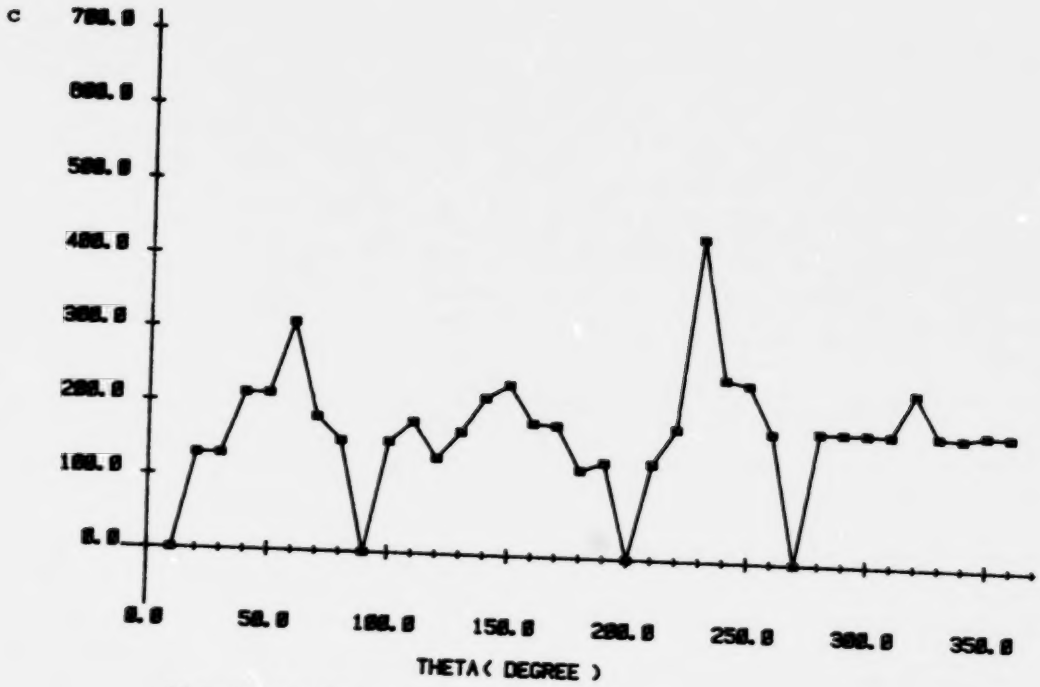


Fig.15 (c) Matching results for TH5 = 0.00001.

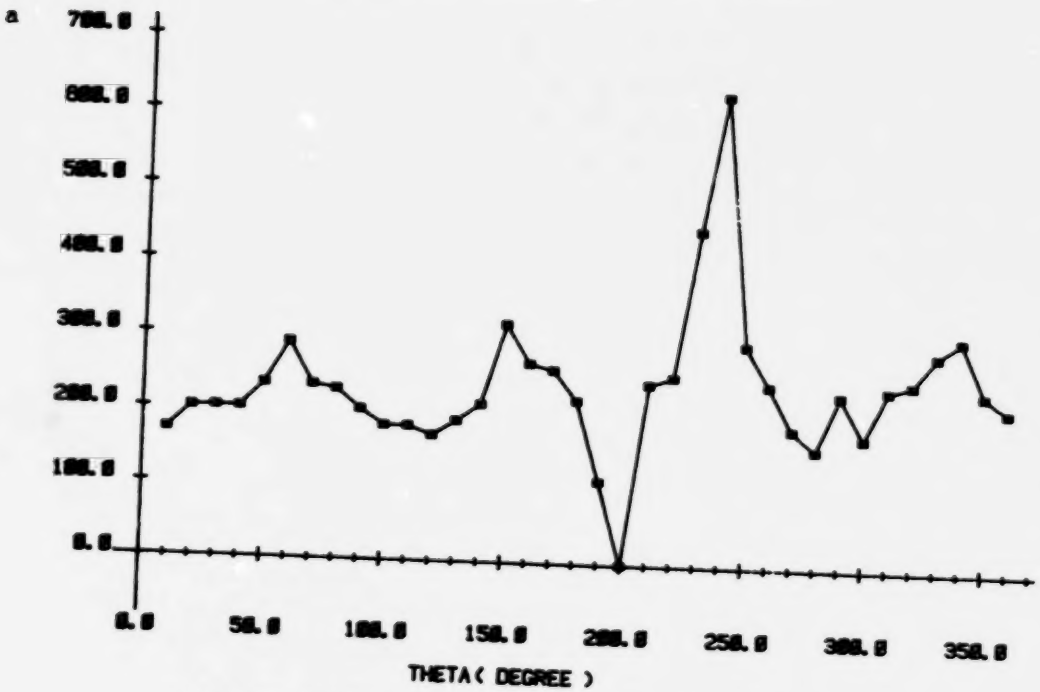


Fig. 16 (a) Matching results for NFD = 2.

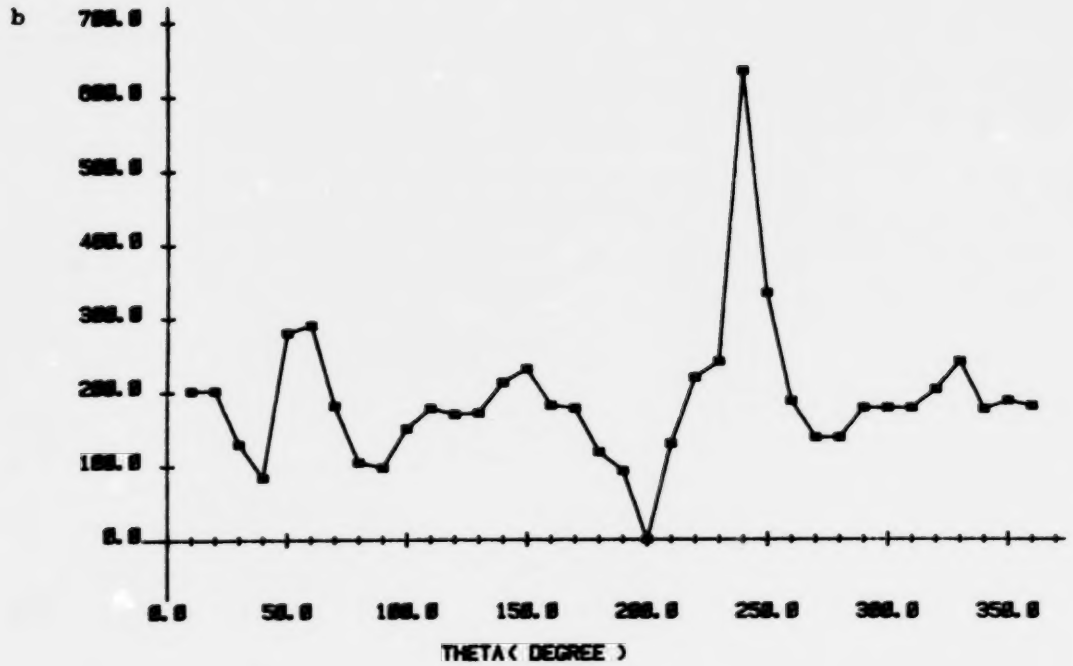


Fig. 16 (b) Matching results for NFD = 4.

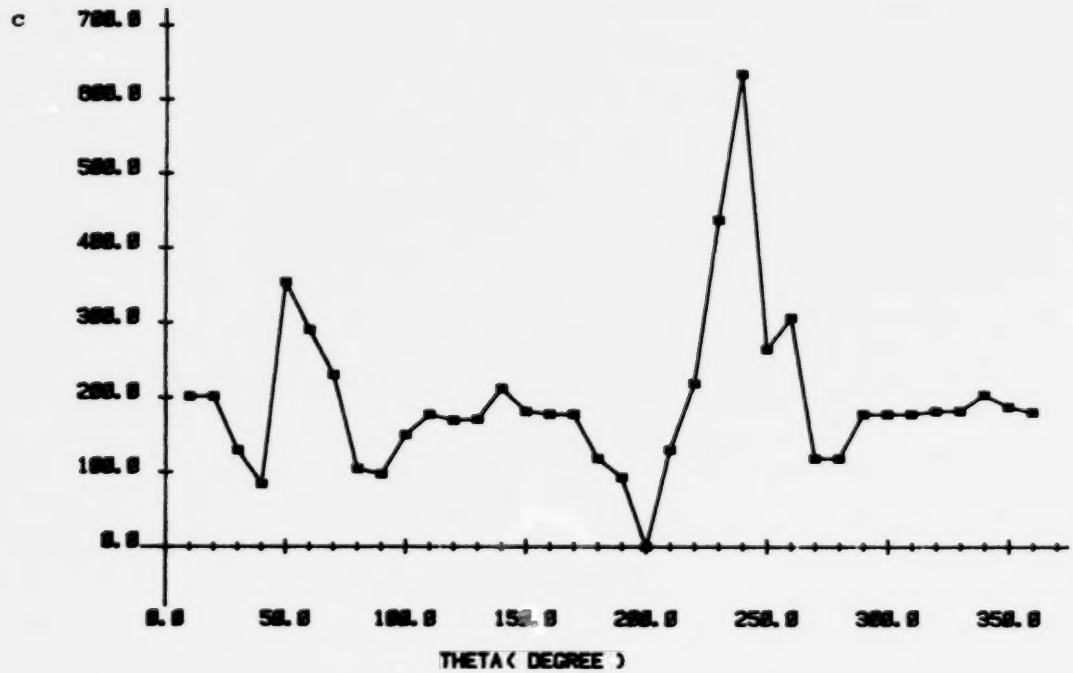


Fig. 16 (c) Matching results for NFD = 5.

Figure 16 shows the results of changing the number NFD (number of terms in the Fourier expansion). The threshold TH5 = 0.00002 was fixed in these cases. The NFDs used were,

NFD = 2 (Fig. 16a)
NFD = 4 (Fig. 16b)
NFD = 5 (Fig. 16c)

The horizontal axis is the position of the model and the vertical axis is the matching score for the best candidate pairs, as described in the previous section. The results show that the 240° position was correctly identified.

5. CONCLUSION

Segment-matching plays an important role in robot vision since we cannot expect normal views for the objects when the environment is irregular. We have presented a new method of segment-matching based on clique detection techniques.

The method utilizes the clique detection technique twice: firstly to detect candidates for matched corner pairs and secondly to merge the consistent sets of pairs detected as different cliques in the first detection. By changing the criterion for connecting edges in the first graph, i.e., in the association graph, the method can be used for both scale-variant and scale-invariant matching problems. In the scale-invariant matching, Fourier descriptors are used as measures for connecting edges between the nodes of the association graph.

The experimental results showed the feasibility of applying this method to real scenes. Though clique detection is a NP-complete problem with respect to its computational complexity, the technique gives us a systematic way of segment-matching of shapes when we can restrict the size of the association graph within practical limits.

ACKNOWLEDGEMENTS

We would like to express our thanks to the director of the Division of Electrical Engineering, Dr. J.Y. Wong, the former Section Head, Dr. M. Wein, and the staff of the Computer Graphics Section for making this study possible.

References

Aho, A., Hopcroft, J., and Ullman, J. The design and analysis of computer algorithms. Addison-Wesley, Reading, MA. 1974.

Ambler, A.P., et al. A versatile system for computer-controlled assembly. Artificial Intelligence, 6: 129-156; 1975.

Bolles, R.C. and Kruger, R.P. Robust feature matching through maximal cliques. Proc. Soc. Photo-Opt. Instrum. Eng. 182: 140-149; 1979.

Bray, D.W. Dynamic storage routine for Fortran programs. SIGPLAN Sept., 1974.

Bron, C., and Kerbosch, J. Algorithm 457: Finding all cliques of an undirected graph. Commun. ACM, 16: 575-577; 1973.

Cheng, J.K., and Huang, T.S. Image recognition by matching relational structures. Proc. IEEE Pattern Recognition and Image Processing Conf. Dallas, TX. 1981. 542-547.

Davis, L.S. Shape matching using relaxation techniques. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1(1): 60-72; 1979.

Hung, S.H.Y., and Kasvand, T. Critical points on a perfectly 8- or 6-connected thin binary line. Proc. 6th Int. Joint Conf. on Pattern Recognition, Munich. 1982. pp. 531-534.

Gerhards, L., and Lindenberg, W. Clique detection for nondirected graphs: Two new algorithms. Computing, 21: 295-322; 1979.

Granlund, G.H. Fourier preprocessing for hand print character recognition. IEEE Trans. Commun. C-21: 195-201; 1972.

Johnston, H.C. Clique of a graph - Variations of the Bron-Kerbosch algorithm. Int. J. Comput. Inf. Sci. 5(3): 209-238; 1976.

Otsu, N. A threshold selection method from gray-level histogram. IEEE Trans. Syst., Man Cybern. SMC-9: 62-66; 1979.

Pavlidis, T., and Ali, F. A hierarchical syntactic shape analyzer. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1(1): 2-9; 1979.

Pavlidis, T. The use of syntactic shape analyzer for contour matching. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1(3): 307-310; 1979.

Pavlidis, T. Algorithms for shape analysis of contours and waveforms. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-2(4): 301-312; 1980.

Perkins, W.A. A model-based vision system for industrial parts. IEEE Trans. Commun. C-27(2): 126-143; 1978.

Persoon, E., and Fu, K.S. Shape discrimination using Fourier descriptors. IEEE Trans. Syst., Man Cybern. SMC-7(3): 170-179; 1977.

Radig, B., Kraasch, R., and Zach, W. Matching symbolic description for 3-D reconstruction of simple moving objects. Proc. 5th Int. Joint Conf. on Pattern Recognition, 1980. pp. 1081-1084.

Ramar, U. An iterative procedure for the polygonal approximation of plane curves. Comput. Graphics Image Process. 1: 244-246; 1972.

Richard, C.W., and Hemami, H. Identification of three-dimensional objects using Fourier descriptors of the boundary curves. IEEE Trans. Syst., Man Cybern. SMC-4(4): 371-378; 1974.

Rutkowski, W.S., Peleg, S., and Rosenfeld, A. Shape segmentation using relaxation. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-3(4): 368-375; 1981.

Rutkowski, W.S. Shape segmentation using relaxation, III. TR-1038, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD. April 1981.

Sakane, S. An algorithm for scale invariant segment-matching. Proc. IEEE Pattern Recognition and Image Processing Conf., Dallas, TX. 1981. pp. 565-571.

Wallace, T.P., Mitchell, O.R., and Fukunaga, K. Three-dimensional shape analysis using local shape descriptors. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-3(3): 310-323; 1981.

You, K.C., and Fu, K.S. A syntactic approach to shape recognition using attributed grammars. IEEE Trans. Syst., Man Cybern SMC-9(6): 334-345; 1979.

You, K.C., and Fu, K.S. Distorted shape recognition using attributed grammars and error-correcting techniques. Comput. Graphics Image Process. 13: 1-13; 1980.

Zahn, C.T., and Roskies, R.Z. Fourier descriptors for plane closed curves. IEEE Trans. Comput. C-21: 269-281; 1972.

APPENDIX A: CALCULATION OF FOURIER DESCRIPTORS

The Fourier expansion of boundary curves has been developed as a global shape measure. Two kinds of methods have been proposed. One is the Fourier expansion of a function which is defined as a tangent angle change versus arc length (Zahn and Roskies 1972). The other method is defined in terms of a complex function Z(t) in which a curve is represented as Z(t) = X(t) + jY(t), where (X(t), Y(t)) is the parametric expression of the curve (Granlund 1972, Richard and Hemami 1974, Persoon and Fu 1977). Since the latter method has no discontinuity in the function, even if the shape is represented as a polygonal curve, the magnitudes of the descriptors decrease faster than in the former method. We employed the descriptors of the latter definition to calculate a similarity measure for two shape segments. The Fourier descriptor is defined as,

$$A(n) = \frac{1}{T^2}\int_0^T Z(t)\exp(-2\pi nt/T)dt, \qquad n = 1, 2, 3,...$$

where T is the total length of a curve and t is the parameter of length from a starting point. The absolute value of A(n), except for n = 0, is invariant in translation, rotation and scale change of the curve. Since a segment of a contour (shape) is not a closed curve, we calculate the Fourier descriptors starting from one endpoint of the segment to the other endpoint and then retrace the curve to the start point to create a closed curve. When we have a segment consisting of m corners from N_1 to N_m and the total length of the segment is L, the Fourier descriptor is calculated as,

$$A(n) = \frac{1}{(2L)^2}\int_0^{2L} Z(t)\exp(-2\pi nt/2L)dt$$

Since Z(t) = Z(2L - t),

$$A(n) = \frac{1}{(2L)^2}\int_0^{L} Z(t)\exp(-\pi nt/L)dt + \frac{1}{(2L)^2}\int_L^{2L} Z(2L-t)\exp(-\pi nt/L)dt$$

We substitute (2L - t) for t in the second term. Then,

$$A(n) = \frac{1}{(2L)^2}\int_0^{L} Z(t)\cos(\pi nt/L)dt$$

When the curve is polygonal as in Fig. A1 and a corner P_i has the position vector V(i), the integration can be calculated by;

$$A(n) = \sum_{i=1}^{m-1} \frac{V(i+1) - V(i)}{2(\pi n)^2\left|V(i+1)-V(i)\right|}\left(\cos(\pi nt_{i+1}/L) - \cos(\pi nt_i/L)\right)$$

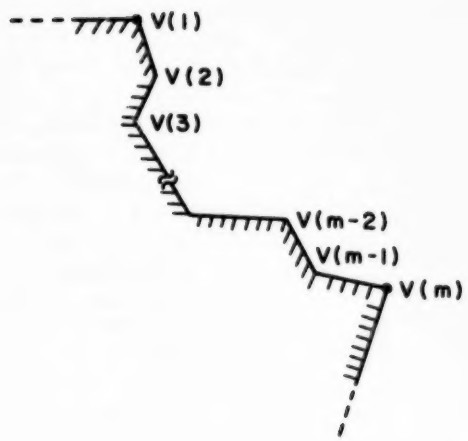
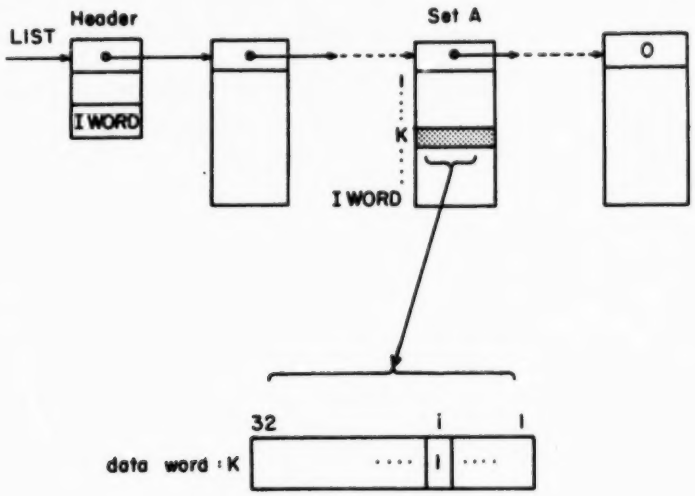


Fig. A1 Position vectors of corners.

APPENDIX B: IMPLEMENTATION OF A CLIQUE DETECTION PROGRAM

We describe the program of clique detection used in the matching program. The program is written in FORTRAN for VAX11/780. A list program package LPN (Bray 1974) is used to manage a stack instead of using the recursive fuction EXPAND. Since most of the important operations in our algorithm are expressed in terms of logical AND, OR, NOT, etc. on sets, a family of routines is introduced for writing compact programs for these operations. To make the data structure efficient, a binary expression for a set is employed (Fig. B1). Bit positions are assigned to all possible elements of a set. An element is regarded as a member of a set if the corresponding bit position contains '1'. Therefore, the logical operations on sets can be implemented efficiently using the functions: IAND, IOR, NOT.



element : ((K-1) * 32 + i) ∈ set A

Fig. B1 Binary representation of sets.

An undirected graph is represented by a table of pointers to the sets of adjacent nodes (Fig. B2). Figure B3 shows the subroutine CLIQ of clique detection. INEIBR is the table of the input graph. ISIZE is the total number of nodes in the graph. ISW is a switch to select the simplified Bron-Kerbosch algorithm and the generalized Bron-Kerbosch algorithm. INISET is an item for a set which is copied to the initial POSSIBLE and INTERSECTION. Since the algorithm regards an isolate node as a clique, it is efficient to remove the isolate nodes from the initial set, if desired, by setting '0' in the bit position of that node. ISZMIN is the minimum size of a clique to be stored in the list JCLIQ. In the experiments of segment-matching, ISZMIN=3 was used. The output JCLIQ is a

list of pointers to the clique sets. Figure B4 shows how the inner stack works in detecting cliques for Fig. 1. In this example, the simplified Bron-Kerbosch algorithm was selected. The order of detecting cliques is different from the original algorithm since the order of stacking is different. The indentations correspond to the levels of the recursive calls of the original algorithm.

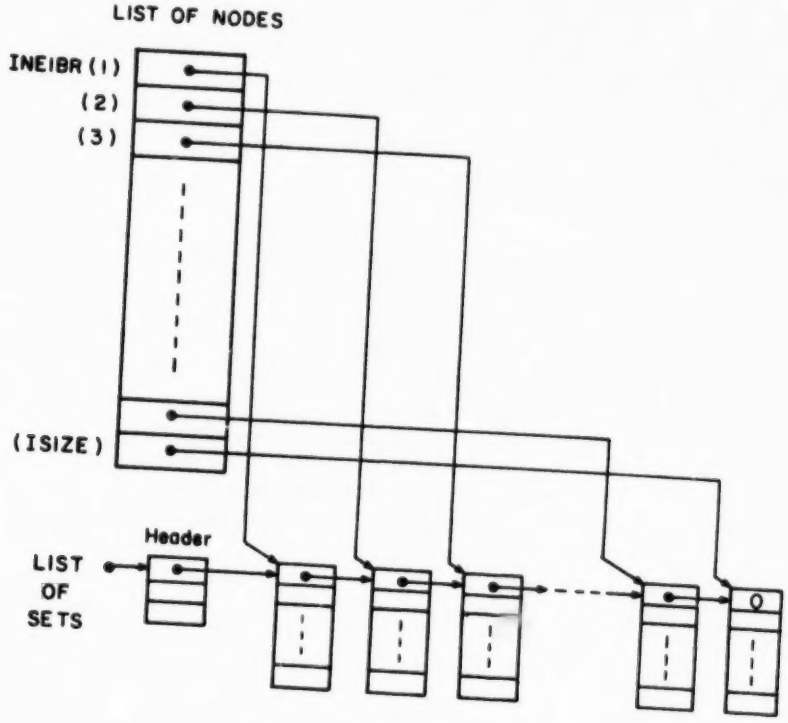


Fig. B2 Format of an undirected graph

```
C
CP      PROGRAMS OF CLIQUE DETECTION IN A GRAPH
CP      BASED ON JOHNSTON'S ALGORITHM
C
CD              1982.3.23       S.SAKANE
C
C-------------------------------------------------------------
C
      SUBROUTINE CLIQ( INEIBR,ISIZE,IWORK,ISW,ITHR,
     &                                    INISET,ISZMIN,JCLIQ )
C
CP      CLIQUE DETECTION IN A GRAPH BASED ON
CP      JOHNSTON'S ALGORITHM
C
CA   INEIBR(ISIZE)  : TABEL OF POINTERS TO THE ADJACENT      (IN)
CA                    NODES OF NODE IN THE GRAPH
CA   ISIZE      : TOTAL NUMBER OF NODES IN THE GRAPH         (IN)
CA   IWORK(ISIZE) : WORK ARRAY                             (WORK)
CA   ISW        : SWITCH OF ALGORITHM SELECTION             (IN)
CA                  =0 --- SIMPLIFIED BRON-KERBOSCH ALGORITHM
CA                  =1 --- GENERALIZED BRON-KERBOSCH ALGORITHM
CA   ITHR       : THRESHOLD FOR GENERALIZED                 (IN)
CA                BRON-KERBOSCH ALGORITHM
CA   INISET     : INITIAL SET OF 'POSSIBLE'                 (IN)
CA   ISZMIN     : MINIMUM SIZE OF CLIQUE TO BE STORED       (IN)
CA   JCLIQ      : LIST OF CLIQUES                          (OUT)
C
CN   INPUT GRAPH HAS A BIT PATTERN FORMAT FOR EDGES OF A NODE.
C
CN   [ REFERENCE ]
CN    H.C.JOHNSTON,'CLIQUES OF A GRAPH - VARIATIONS ON THE BRON-
CN    KERBOSCH ALGORITHM,'  INTERNATIONAL JOURNAL OF COMPUTER
CN    AND INFORMATION SCIENCES, VOL.5, NO.3, PP.209-238, 1976.
C
      DIMENSION INEIBR(1),IWORK(1)
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/JSIZE,IWORD,MCNBIT,MASK(32)
      COMMON /TMPSET/IPT1,IPT2
C
      CALL NSINIT( ISIZE )
C
CF *** MAKE TWO SET ITEMS FOR BUFFER IN 'CLIQI1' AND 'CLIQI2'
C
      ITMP = NEWLST( IWORD )
      IPT1 = NEWTOP( ITMP,1,0 )
      IF( ISW.EQ.0 ) GO TO 10
      IPT2 = NEWTOP( ITMP,1,0 )
   10 CONTINUE
```

Fig. B3 Program list for clique detection.

```
C
CF *** INITIALIZE THE LISTS 'ISET','ISTACK','JCLIQ'
C
      ISET = NEWLST( IWORD )
      ISTACK = NEWLST( 3 )
      JCLIQ  = NEWLST( 1 )
C
CF *** INITIALLIZE THE SETS;
CF      'COMBINATION','INTERSECTION','POSSIBLE'.
C
      KPT1 = NEWTOP( ISET,1,0 )
      KPT2 = NEWTOP( ISET,1,0 )
      KPT3 = NEWTOP( ISET,1,0 )
      CALL NSEMPT( KPT1 )
      CALL NSCOPY( INISET, KPT2 )
      CALL NSCOPY( INISET, KPT3 )
C
CF *** PUT THE FIRST SETS ON TOP OF THE STACK
C
      NPT = NEWTOP( ISTACK,1,0 )
      LSP( NPT+1 ) = KPT1
      LSP( NPT+2 ) = KPT2
      LSP( NPT+3 ) = KPT3
C
CF *** REPEAT UNTIL THE STACK IS EMPTY.
C
  100 CONTINUE
        ITEM = NPOP( ISTACK, 1, KCOM )
        IF( ITEM.EQ.0 ) GO TO 900
        KINT = LSP( ITEM + 2 )
        KPOS = LSP( ITEM + 3 )
C
CF *** CALCULATE THE SET 'S' ACCORDING THE SWITCH 'ISW'
C
        IF( ISW.EQ.0 ) CALL CLIQI1( INEIBR,KINT,KPOS,IWORK,JNUM )
        IF( ISW.EQ.1 ) CALL CLIQI2( INEIBR,KINT,KPOS,ITHR,
     &                                                IWORK,JNUM )
C
CF *** WHEN 'S' IS EMPTY, GO TO THE PROCESSING OF DEAD END
C
        IF( JNUM.EQ.0 ) GO TO 600
        DO 500 INDEX=1,JNUM
          INODE = IWORK( INDEX )
C
CF *** REMOVE 'INODE' FROM 'POSSIBLE'
C
        CALL NSRMVE( KPOS, INODE )
        KPT1 = NEWTOP( ISET,1,0 )
        CALL NARENT( KPT1,1,IWORD,LSP(KCOM+1) )
```

Fig. B3 (Continued)

```
C
CF *** NEW 'COMBINATION' = 'COMBINATION' U 'INODE'
C
        CALL NSADDE( KPT1, INODE )
C
CF *** NEW 'INTERSECTION' = 'INTERSECTION' .AND. G[ INODE ]
CF *** NEW 'POSSIBLE' = 'POSSIBLE' .AND. G[ INODE ]
CF       WHERE G[N] MEANS THE NEIBOUR NODES OF A NODE 'N'
C
        NPT = INEIBR( INODE )
        KPT2 = NEWTOP( ISET,1,0 )
        KPT3 = NEWTOP( ISET,1,0 )
        IFLAG2 = 0
        IFLAG3 = 0
C
CV *** IFLAG2,IFLAG3 ARE USED TO DETECT EMPTY SET.
C
        DO 300 I=1,IWORD
          MV = LSP( NPT + I )
          IV2 = IAND( LSP( KINT+I), MV )
          IV3 = IAND( LSP( KPOS+I), MV )
          IF( IV2.NE.0 ) IFLAG2=1
          IF( IV3.NE.0 ) IFLAG3=1
          LSP( KPT2+I ) = IV2
          LSP( KPT3+I ) = IV3
  300   CONTINUE
        IF( IFLAG2.EQ.1 ) GO TO 350
C
CF *** NEW 'INTERSECTION' IS EMPTY, A CLIQUE IS FOUND.
C
        CALL NDELET( ISET, KPT2 )
        CALL NDELET( ISET, KPT3 )
C
CF *** CALCULATE THE SIZE OF THE CLIQUE
C
        CALL NSSIZE( KPT1, L )
C
CF *** COMPARE THE SIZE WITH 'ISZMIN'
C
        IF( L.LT.ISZMIN ) GO TO 320
  310   CONTINUE
        CALL NEWTOP( JCLIQ, 1, KPT1 )
        GO TO 500
  320   CONTINUE
        CALL NDELET( ISET, KPT1 )
        GO TO 500
  350   CONTINUE
C
CF *** CHECK IF NEW 'POSSIBLE' IS EMPTY.
```

Fig. B3 (Continued)

```
C
        IF( IFLAG3.EQ.1 ) GO TO 410
C
CF *** DEAD END IS FOUND.
C
        CALL NDELET( ISET, KPT3 )
        CALL NDELET( ISET, KPT2 )
        CALL NDELET( ISET, KPT1 )
        GO TO 500
C
  410   CONTINUE
C
CF *** CREATE A NEW ITEM ON THE STACK FOR BACKTRACKING.
C
        ITEM = NEWTOP( ISTACK, 1, KPT1 )
        LSP( ITEM+2 ) = KPT2
        LSP( ITEM+3 ) = KPT3
C
  500   CONTINUE
        GO TO 100
C
  600   CONTINUE
C
CF *** DEAD END IS FOUND
C
        CALL NDELET( ISET, KINT )
        IF( KPOS.NE.0 ) CALL NDELET( ISET,KPOS )
        IF( KCOM.NE.0 ) CALL NDELET( ISET,KCOM )
        GO TO 100
C
  900 CONTINUE
      CALL NERASE( ITMP )
      CALL NERASE( ISTACK )
      RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE CLIQI1( INEIBR,KINT,KPOS,JTBL,JNUM )
C
CP   CALCULATE THE SET 'S'
CP   ( SIMPLIFIED BRON-KERBOSCH ALGORITHM )
C
CA   INEIBR(ISIZE)  : POINTER TO EDGE ITEM OF NODE        (IN)
CA   KINT           : POINTER TO 'INTERSECTION'           (IN)
CA   KPOS           : POINTER TO 'POSSIBLE'               (IN)
CA   JTBL(JNUM)     : OUTPUT SET 'S'                      (OUT)
CA   JNUM           : SIZE OF 'S'                         (OUT)
```

Fig. B3 (Continued)

```
C
      DIMENSION INEIBR(1),JTBL(1)
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32)
      COMMON /TMPSET/IPT1
C
      IBIAS = 0
      DO 30 J=1,IWORD
        IV = LSP( KINT + J )
        IF( IV.EQ.0 )  GO TO 20
        DO 10 I=1,MCNBIT
          IVAL = IAND( IV, MASK(I) )
          IF( IVAL.NE.0 ) GO TO 40
          IF( (IBIAS+I).EQ.ISIZE ) GO TO 90
   10   CONTINUE
   20   CONTINUE
        IBIAS = IBIAS + MCNBIT
   30 CONTINUE
   40 CONTINUE
      NPT = INEIBR( IBIAS + I )
C
CF *** 'POSSIBLE' .AND. NOT( G[ ANY NODE IN 'INTERSECTION'] )
      CALL NSSUBT( KPOS, NPT, IPT1 )
      CALL NSLIST( IPT1, JTBL,JNUM )
      RETURN
C
   90 CONTINUE
      WRITE(6,*) 'ERROR IN CLIQI1.'
      RETURN
      END



      SUBROUTINE CLIQI2( INEIBR, KINT, KPOS, ITHR, JTBL, JNUM )
C
CP    CALCULATE THE SET 'S'
CP    ( GENERALIZED BRON-KERBOSCH ALGORITHM )
C
CA    INEIBR(ISIZE)    : POINTER TO EDGE ITEM OF NODE      (IN)
CA    KINT             : POINTER TO 'INTERSECTION'         (IN)
CA    KPOS             : POINTER TO 'POSSIBLE'             (IN)
CA    ITHR             : THRESHOLD VALUE                   (IN)
CA    JTBL(JNUM)       : OUTPUT SET 'S'                   (OUT)
CA    JNUM             : SIZE OF 'S'                      (OUT)
```

Fig. B3 (Continued)

```
C
      DIMENSION INEIBR(1),JTBL(1)
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32)
      COMMON /TMPSET/ISPT,NEWS
      DATA MAXVAL/10000000/
C
      KSIZE = MAXVAL
      NUM  = 0
      IBIAS = 0
      DO 2000 J=1,IWORD
        IV = LSP( KINT + J )
        IF( IV.EQ.0 ) GO TO 1500
        DO 1000 I=1,MCNBIT
          NUM = IBIAS + I
          IVAL = IAND( IV, MASK(I) )
          IF( IVAL.EQ.0 ) GO TO 800
          CALL NSSUBT( KPOS,INEIBR(NUM),NEWS )
          CALL NSSIZE( NEWS, NEWSIZ )
          IF( NEWSIZ.GE.KSIZE ) GO TO 800
          CALL NSCOPY( NEWS, ISPT )
          KSIZE = NEWSIZ
          IF( KSIZE.LE.ITHR ) GO TO 5000
  800     CONTINUE
          IF( NUM.EQ.ISIZE ) GO TO 5000
 1000   CONTINUE
 1500 CONTINUE
      IBIAS = IBIAS + MCNBIT
 2000 CONTINUE
 5000 CONTINUE
      CALL NSLIST( ISPT,JTBL,JNUM )
C
 9000 CONTINUE
      RETURN
      END
$


C
CP *** SET OPERATION UTILITY ***
C
CN       A SET CORRESPONDS TO A LIST ITEM IN WHICH THE ELEMENTS
CN       ARE REPRESENTED AS A BIT PATTERN TO USE LOGICAL
CN       OPERATIONS AS 'IAND','IOR','NOT'.
C
CN       LIST PROCESSING PACKAGE 'LPN' IS USED IN THE ROUTNES.
C
CN       SINCE THE INPUT ARGUMENTS OF ROUTINES ARE GENERALLY
CN       POINTERS TO SET ITEMS, THE CONSTRUCTION OF THEIR LIST
CN       HAS TO BE GIVEN IN USER PROGRAM.
```

Fig. B3 (Continued)

```
C
CD      1982.01.14      S.SAKANE
C
C
C       [ LIST OF UTILITY ROUTINES ]
C
C       ( NAME )        ( FUNCTION )
C
C       NSINIT          INITIALIZE
C       NSMEMB          CHECK IF AN ELEMENT IS MEMBER OF A SET
C       NSADDE          ADD AN ELEMENT TO A SET
C       NSRMVE          REMOVE AN ELEMENT FROM A SET
C       NSSIZE          GET THE SIZE OF A SET
C       NSSUBT          SUBTRACT SET (B) FROM SET (A)
C       NSCOPY          COPY SET (A) TO SET (B)
C       NSUNIO          UNION OF SET (A) AND SET (B)
C       NSINTR          INTERSECTION OF SET (A) AND SET (B)
C       NSEMPT          MAKE AN EMPTY SET
C       NSCMPL          MAKE THE COMPLEMENT OF A SET
C       NSFULL          MAKE A SET WITH ALL POSSIBLE ELEMENTS
C       NSLIST          LIST UP ALL ELEMTNTS IN A SET
C       NSINTS          RETURN THE STATUS OF INTERSECTION
C
C-------------------------------------------------------------------
C
      SUBROUTINE NSINIT( ISIZE )
C
CP      INITIALIZE PARAMETERS OF SET UTILITIES
C
CS      CALL NSINIT( ISIZE )
C
CA      ISIZE : MAXIMUM NUMBER OF ELEMENTS IN A SET      (IN)
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/JSIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
CV *** MCNBIT, MASK, MSKFUL ARE MACHINE DEPENDENT PARAMETERS.
CV      ( A MACHINE WITH 32 BITS/WORD IS ASSUMED HERE )
C
      DATA MCNBIT/32/
      DATA MSKFUL/'FFFFFFFF'X/
C
      JSIZE = ISIZE
      IWORD = ( ISIZE-1 )/MCNBIT + 1
C
      IVAL = 1
      DO 10 I=1,MCNBIT
```

Fig. B3 (Continued)

```
          MASK( I ) = IVAL
          IVAL = ISHFT( IVAL, 1 )
       10 CONTINUE
          RETURN
          END
C
C-----------------------------------------------------------------
C
      SUBROUTINE NSMEMB( ISET, IELM, JVAL )
C
CP      CHECK IF AN ELEMENT IS INCLUDED IN A SET
C
CS      CALL NSMEMB( ISET, IELM, JVAL )
C
CA      ISET     : POINTER TO SET ITEM            (IN)
CA      IELM     : ELEMTNT TO BE CHECKED          (IN)
CA      JVAL     : =0 --- NOT INCLUDED            (OUT)
CA                 =1 --- INCLUDED
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      KWORD = ( IELM - 1 )/MCNBIT + 1
      KBIT  = MOD( IELM-1, MCNBIT ) + 1
      IF( KBIT.EQ.0 ) KBIT=MCNBIT
C
      JVAL  = 0
      K = IAND( LSP( ISET+KWORD ), MASK( KBIT ) )
      IF( K.NE.0 ) JVAL=1
      RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE NSADDE( ISET, IELM )
C
CP      ADD AN ELEMENT TO A SET
C
CS      CALL NSADDE( ISET, IELM )
C
CA      ISET     : POINTER TO A SET ITEM          (IN/OUT)
CA      IELM     : ELEMENT TO ADD                 (IN)
C
        COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      KWORD = ( IELM - 1 )/MCNBIT + 1
      KBIT  = MOD( IELM-1, MCNBIT ) + 1
```

Fig. B3 (Continued)

```
C
      LSP( ISET+KWORD ) = IOR( LSP(ISET+KWORD), MASK(KBIT) )
      RETURN
      END
C
C-------------------------------------------------------------
C
      SUBROUTINE NSRMVE( ISET, IELM )
C
CP      REMOVE AN ELEMENT FROM A SET
C
CS      CALL NSRMVE( ISET, IELM )
C
CA      ISET    : POINTER TO SET ITEM            (IN/OUT)
CA      IELM    : ELEMENT TO REMOVE              (IN)
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      KWORD = ( IELM - 1 )/MCNBIT + 1
      KBIT  = MOD( IELM-1, MCNBIT ) + 1
C
      KMASK = NOT( MASK( KBIT ) )
      LSP(ISET+KWORD) = IAND( LSP(ISET+KWORD), KMASK )
      RETURN
      END
C
C-------------------------------------------------------------
C
      SUBROUTINE NSSIZE( ISET, JSIZE )
C
CP      CALCULATE THE SIZE OF A SET
C
CS      CALL NSSIZE( ISET, JSIZE )
C
CA      ISET    : POINTER TO SET ITEM            (IN)
CA      JSIZE   : SIZE OF SET                    (OUT)
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      KCOUNT = 0
      INUM   = 0
C
      DO 10 I=1,IWORD
        IDATA = LSP( ISET+I )
        DO 20 KBIT = 1,MCNBIT
```

Fig. B3 (Continued)

```
            INUM = INUM + 1
            IVAL = IAND( IDATA, MASK(KBIT) )
            IF( IVAL.NE.0 ) KCOUNT=KCOUNT+1
            IF( INUM.EQ.ISIZE ) GO TO 90
         20 CONTINUE
         10 CONTINUE
C
         90 CONTINUE
            JSIZE = KCOUNT
            RETURN
            END
C
C-------------------------------------------------------------------
C
      SUBROUTINE NSSUBT( ISETA, ISETB, JSETC )
C
CP      SUBTRACTION : (C) = (A) - (B)
C
CS      CALL NSSUBT( ISETA, ISETB, JSETC )
C
CA      ISETA   : POINTER TO SET (A)                    (IN)
CA      ISETB   : POINTER TO SET (B)                    (IN)
C       ISETC   : POINTER TO SET (C)                    (OUT)
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      DO 10 J=1,IWORD
        LSP( JSETC+J) = IAND( LSP(ISETA+J),NOT(LSP(ISETB+J)) )
   10 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------------
C
      SUBROUTINE NSCOPY( ISETA, JSETB )
C
CP      COPY SET (A) TO SET (B)
C
CS      CALL NSCOPY( ISETA, JSETB )
C
CA      ISETA   : POINTER TO SET (A)                    (IN)
CA      JSETB   : POINTER TO SET (B)                    (OUT)
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      DO 10 I=1,IWORD
```

Fig. B3 (Continued)

```
        LSP( JSETB+I ) = LSP( ISETA+I )
   10 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
C
      SUBROUTINE NSUNIO( ISETA, ISETB, JSETC )
C
CP      UNION : SET (C) = (A) U (B)
C
CS      CALL NSUNIO( ISETA, ISETB, JSETC )
C
CA      ISETA    : POINTER TO SET (A)            (IN)
CA      ISETB    : POINTER TO SET (B)            (IN)
CA      JSETC    : POINTER TO SET (C)            (OUT)
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      DO 10 I=1,IWORD
        LSP( JSETC+I ) = IOR( LSP( ISETA+I ), LSP( ISETB+I ) )
   10 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
C
      SUBROUTINE NSINTR( ISETA, ISETB, JSETC )
C
CP      INTERSECTION : SET (C) = (A) .AND. (B)
C
CS      CALL NSINTR( ISETA, ISETB, JSETC )
C
CA      ISETA    : POINTER TO SET (A)            (IN)
CA      ISETB    : POINTER TO SET (B)            (IN)
CA      JSETC    : POINTER TO SET (C)            (OUT)
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      DO 10 I=1,IWORD
        LSP( JSETC+I ) = IAND( LSP( ISETA+I ), LSP( ISETB+I ) )
   10 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
```

Fig. B3 (Continued)

```
C
      SUBROUTINE NSEMPT( ISET )
C
CP      MAKE AN EMPTY SET
C
CS      CALL NSEMPT( ISET )
C
CA      ISET    : POINTER TO SET          (IN/OUT)
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      DO 10 I=1,IWORD
        LSP( ISET+I ) = 0
   10 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
C
      SUBROUTINE NSCMPL( ISETA, JSETB )
C
CP      COMPLEMENT  : (B) = COMPLEMENT OF (A)
C
CS      CALL NSCMPL( ISETA, JSETB )
C
CA      ISETA   : POINTER TO SET (A)     (IN)
CA      JSETB   : POINTER TO SET (B)     (OUT)
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      DO 10 I=1,IWORD
        LSP( JSETB+I ) = NOT( LSP( ISETA+I ) )
   10 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
C
      SUBROUTINE NSFULL( ISET )
C
CP      MAKE A SET WITH ALL POSSIBLE ELEMENTS
C
CS      CALL NSFULL( ISET )
C
CA      ISET    : POINTER TO SET          (IN/OUT)
C
```

Fig. B3 (Continued)

```
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      DO 10 I=1,IWORD
        LSP( ISET+I ) = MSKFUL
   10 CONTINUE
C
      KBIT = MOD( ISIZE, MCNBIT )
      IF( KBIT.EQ.0 ) GO TO 20
C
      LSP( ISET+IWORD ) = ISHFT( MSKFUL, KBIT-MCNBIT )
   20 CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE NSLIST( ISET, JLIST, JSIZE )
C
CP      LIST UP THE ELEMENTS OF A SET
C
CS      CALL NSLIST( ISET, JLIST, JSIZE )
C
CA      ISET            : POINTER TO SET ITEM          (IN)
CA      JLIST(JSIZE)    : LIST OF ELEMENTS             (OUT)
CA      JSIZE           : SIZE OF SET                  (OUT)
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
      DIMENSION JLIST(1)
C
      KCOUNT = 0
      INUM   = 0
C
      DO 10 I=1,IWORD
        IDATA = LSP( ISET+I )
        DO 20 KBIT=1,MCNBIT
          INUM = INUM + 1
          IVAL = IAND( IDATA, MASK(KBIT) )
          IF( IVAL.EQ.0 ) GO TO 15
          KCOUNT = KCOUNT + 1
          JLIST( KCOUNT ) = INUM
   15     CONTINUE
          IF( INUM.EQ.ISIZE ) GO TO 90
   20   CONTINUE
   10 CONTINUE
```

Fig. B3 (Continued)

```
C
   90 CONTINUE
      JSIZE = KCOUNT
      RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE NSINTS( ISETA, ISETB, JSTAT )
C
CP      RETURN THE STATUS OF INTERSECTION BETWEEN (A) AND (B)
C
CA      ISETA    : POINTER TO SET ITEM (A)        (IN)
CA      ISETB    : POINTER TO SET ITEM (B)        (IN)
CA      JSTAT    : STATUS OF INTERSECTION        (OUT)
CA                 =0 --- INTERSECTION IS EMPTY.
CA                 =1 --- INTERSECTION IS NOT EMPTY.
C
      COMMON /AVSP/LSP(10)
      COMMON /SETPRM/ISIZE,IWORD,MCNBIT,MASK(32),MSKFUL
C
      JSTAT = 0
      DO 10 K=1,IWORD
        IV = IAND( LSP(ISETA+K), LSP(ISETB+K) )
        IF( IV.NE.0 ) GO TO 80
   10 CONTINUE
      RETURN
   80 CONTINUE
      JSTAT = 1
      RETURN
      END
$
```

Fig. B3 (Concluded)

```
POP UP (COM,INT,POS)
 ( NIL )
 (   1  2  3  4  5  6  7  8  9 )
 (   1  2  3  4  5  6  7  8  9 )
S=(  1  4  5  6  7  8 )
I=  1 EXPAND.
I=  4 EXPAND.
I=  5 EXPAND.
I=  6 EXPAND.
I=  7 EXPAND.
I=  8 EXPAND.
      POP UP (COM,INT,POS)
       (   8 )
       (   4  5  6  7  9 )
       (   9 )
      S=(  9 )
      I=  9 DEAD END 1.
      POP UP (COM,INT,POS)
       (   7 )
       (   6  8  9 )
       (   8  9 )
      S=( NIL )
      I=  9 DEAD END 2.
      POP UP (COM,INT,POS)
       (   6 )
       (   4  7  8  9 )
       (   7  8  9 )
      S=(  7  9 )
      I=  7 EXPAND.
      I=  9 EXPAND.
            POP UP (COM,INT,POS)
             (   6  9 )
             (   7  8 )
             (   8 )
            S=( NIL )
            I=  9 DEAD END 2.
            POP UP (COM,INT,POS)
             (   6  7 )
             (   8  9 )
             (   8  9 )
            S=(  8 )
            I=  8 EXPAND.
                  POP UP (COM,INT,POS)
                   (   6  7  8 )
                   (   9 )
                   (   9 )
                  S=(  9 )
                  I=  9   CLIQUE IS FOUND.
                   (   6  7  8  9 )
$
```

Fig. B4 An example of detecting cliques of Fig. 1.

```
POP UP (COM,INT,POS)
 (    5 )
 (    8   9 )
 (    8   9 )
S=(   8 )
I=   8 EXPAND.
       POP UP (COM,INT,POS)
        (    5   8 )
        (    9 )
        (    9 )
       S=(   9 )
       I=   9    CLIQUE IS FOUND.
        (    5   8   9 )
POP UP (COM,INT,POS)
 (    4 )
 (    6   8 )
 (    6   8 )
S=(   6 )
I=   6 EXPAND.
       POP UP (COM,INT,POS)
        (    4   6 )
        (    8 )
        (    8 )
       S=(   8 )
       I=   8    CLIQUE IS FOUND.
        (    4   6   8 )
POP UP (COM,INT,POS)
 (    1 )
 (    2   3   9 )
 (    2   3   9 )
S=(   2   9 )
I=   2 EXPAND.
I=   9    CLIQUE IS FOUND.
 (    1   9 )
       POP UP (COM,INT,POS)
        (    1   2 )
        (    3 )
        (    3 )
       S=(   3 )
       I=   3    CLIQUE IS FOUND.
        (    1   2   3 )
$
```

Fig. B4 (Concluded)

POOR

QUALITY

ORIGINAL